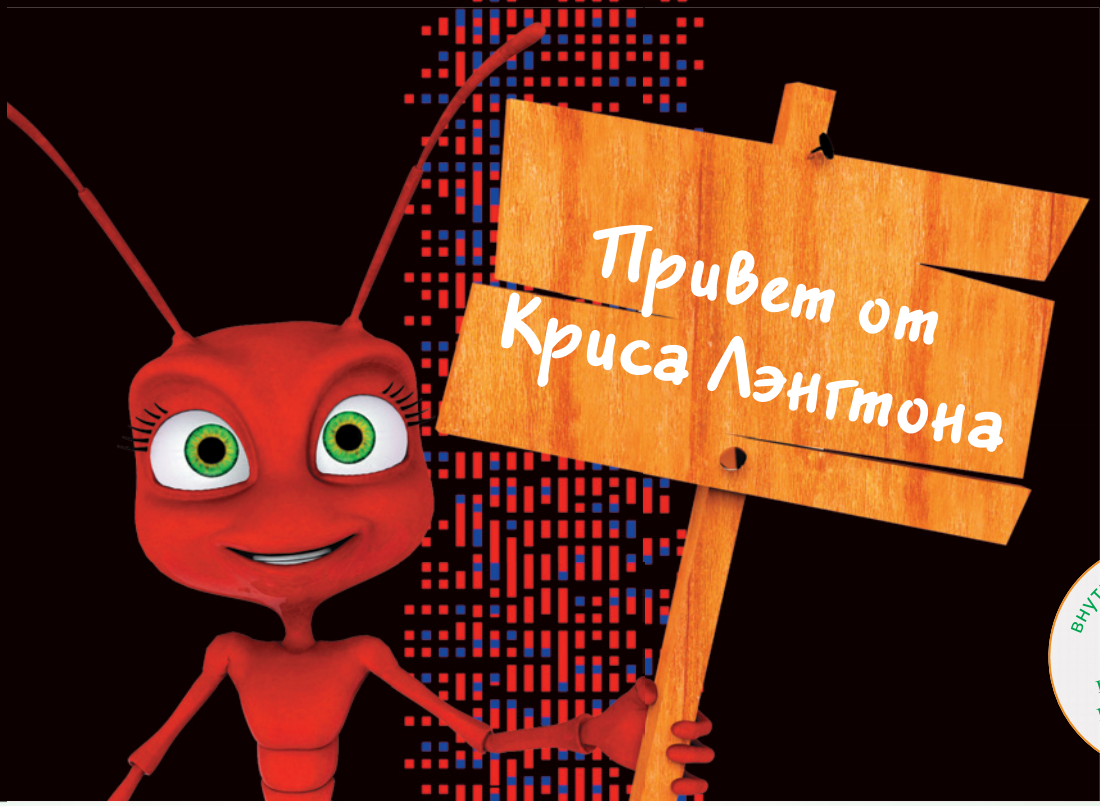
№5 Издание основано в 1995 г. inf.1september.ru
УЧЕБНО-МЕТОДИЧЕСКИЙ ЖУРНАЛ ДЛЯ УЧИТЕЛЕЙ ИНФОРМАТИКИ

ИНФОРМАТИКА



внутри номера CD и код доступа к электронной версии

издательский дом 1september.ru **Первое сентября** | май 2013
ИНФОРМАТИКА Подписка: «Роспечать» – 32291 (бумажная версия), 19179 (электронная); «Почта России» – 79066 (бумажная версия), 12684 (электронная)

НА ОБЛОЖКЕ

Про клеточный автомат “Игра Жизнь” знают все, кто имеет отношение к информатике. Менее известен другой интересный клеточный автомат — “Муравей Лэнгтона”, придуманный биологом Крисом Лэнгтоном. Как и у “Жизни”, здесь тоже есть клетки, но сами по себе они не живут, а раскрашиваются ползающим по полю муравьем. Задачка не сложная, красивая и допускающая много обобщений. Попробуйте!

В НОМЕРЕ

НА ДИСКЕ

ЭЛЕКТРОННЫЕ МАТЕРИАЛЫ:

- Презентации к статьям номера

ИНФОРМАТИКА

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

http://inf.1september.ru

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

РЕДАКЦИЯ:
гл. редактор С.Л. Островский
редакторы
Е.В. Андреева,
Д.М. Златопольский
(редактор вкладки “В мир информатики”)
Дизайн макета И.Е. Лукьянов
верстка Н.И. Пронская
корректор Е.Л. Володина
секретарь Н.П. Медведева
Фото: фотобанк Shutterstock
Журнал распространяется по подписке
Цена свободная
Тираж 25 478 экз.
Тел. редакции: (499) 249-48-96
E-mail: inf@1september.ru
http://inf.1september.ru

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

ГАЗЕТА ИЗДАТЕЛЬСКОГО ДОМА
Первое сентября – Е.Бирюкова

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

УЧРЕДИТЕЛЬ:
ООО “ЧИСТЫЕ ПРУДЫ”

Зарегистрировано ПИ № ФС77-44341 от 22.03.2011
в Министерстве РФ по делам печати
Подписано в печать:
по графику 10.04.2013,
фактически 10.04.2013
Заказ №
Отпечатано в ОАО “Первая Образцовая типография”
Филиал “Чеховский Печатный Двор”
ул. Полиграфистов, д. 1,
Московская область,
г. Чехов, 142300
Сайт: www.chpd.ru
E-mail: sales@chpk.ru
Факс: 8 (495) 988-63-76

АДРЕС ИЗДАТЕЛЯ:
ул. Киевская, д. 24,
Москва, 121165
Тел./факс: (499) 249-31-38

Отдел рекламы:
(499) 249-98-70
http://1september.ru

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

Капитан “Очевидность”

Мне повезло — в моем окружении много молодых талантливых коллег, которые “горят”, “хотят”, пробуют, бьются, терпят неудачи и одерживают победы. Наблюдать за ними и в меру надобности и сил помогать им — большое удовольствие. И в процессе получения этого удовольствия часто приходится решать достаточно типовую мета-задачу, о которой мне хочется сегодня рассказать.

В общем виде запросы, с которыми ко мне нередко обращаются коллеги, выглядят так: “Как ты это делаешь? Почему ты это делаешь? Почему ты делаешь это так, а не иначе? Почему один раз ты сделал так, а другой — иначе?”. Узнаете? Уверен, что и к вам такие вопросы были обращены много раз. В каждом конкретном случае на большинство таких вопросов можно дать достаточно определенный ответ. Но вспомните, не возникало ли у вас ощущение, что ваш четкий, понятный, нередко иллюстрированный примерами ответ как бы “не впрок”. Что через некоторое время коллега снова приходит с вопросом о том же. Или о похожем. Было ведь?

У меня — было. И дело вовсе не в том, что учить и помогать “ломает” или надоедает. Напротив — помогать хочется, это тебе самому доставляет удовольствие, но низкий КПД собственной деятельности заставляет задуматься, что и как надо сделать иначе.

Мне самому далеко ходить не пришлось. Я постарался проанализировать то, как сам учился (и учусь) у своих замечательных учителей, коллег, друзей. Мысль, к которой я пришел, оказалась такой простой, что я уже не уверен, что стоит все это излагать ☺. Впрочем, раз начал…

Вывод такой: я ничего не делаю так же, как те, у кого я этому научился. Более того, я ничего не делаю по тем же основаниям, как они. Я все делаю по-своему. Работа мастеров, которую я наблюдал и наблюдаю, лишь запускает мою собственную внутреннюю работу по пониманию и преломлению их методов, приемов, их стиля.

Важно, что это понимание ничего не отменяет! Все равно важно наблюдать. Все равно важно спрашивать. Все равно важно получать ответы на вопросы “почему?” и “как?”. Все это совершенно необходимо, но только для того, чтобы у каждого из нас непрерывно кипело и пыхтело собственное, личное варево из умений, навыков, мыслей, чувств, идей.

Еще раз повторюсь: это очень простое понимание. И то, что у меня самого его некоторое время не было, — это, вполне возможно, моя личная проблема. Я помню, как стремился сделать “как гуру”, а не получалось ничего. Я иногда расстраивался. А потом в какой-то момент делал “как я”. И у меня получалось. И лишь потом я понял, что получалось именно “как у гуру”, но только “как я”.

Кстати, я знаю, когда пришло ко мне это понимание. Это было после семинаров гениального Фрэнка Фаррелли — автора методов провокативной юморотерапии. Его работа бесподобна. Но каждую секунду семинаров я понимал, что именно так я точно не могу. “Так” — это противоречит моей натуре. Прошло совсем немного времени, и я отметил, что сам (не без удовольствия, возможно даже не без изящества ☺) использую методы Фрэнка. Но только использую их “как я”, от имени себя, от своей натуры, от своего нутра. Я научился у Фрэнка, не копируя буквально ни одного его приема. И боюсь даже подумать, что ответил бы Фрэнк на вопрос “Почему ты сделал это так, а не иначе?” ☺. Но это был бы его ответ, не мой.

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

Имитационное моделирование

И.А. Калинин,
к. п. н., доцент кафедры информатики и прикладной математики МГПУ

Н.Н. Самылкина,
к. п. н., доцент, профессор кафедры теории и методики обучения информатике, МПГУ

Этой статьей мы открываем небольшой цикл публикации материалов нового углубленного курса информатики, представленного в УМК авторов И.А. Калинина и Н.Н. Самылкиной.

УМК состоит из:

1. Программы по информатике, старшая школа, углубленный уровень.
2. Учебника для 10-го класса (в Федеральном перечне учебников с 2013 года).
3. Учебника для 11-го класса (в Федеральном перечне учебников с 2013 года).
4. Задачника-практикума с диском.
5. Сборника проверочных тестов.
6. Методического пособия для учителя.

Авторы отобрали для публикации совершенно новый для курса информатики содержательный материал по темам: “Основы имитационного моделирования”, “Технология обработки текста (регулярные выражения, частотный анализ)”, “Обработка звука”, “Технология обработки графики”.

В этом номере журнала представлены первые две темы из перечисленных.

Практические работы предваряет теоретический материал в минимально необходимом объеме. Для выполнения практических работ необходимо установить рекомендуемую к использованию программную среду с сайта компании The AnyLogic Company. Продукт платный, но всем доступна демонстрационная версия программного продукта для использования в течение месяца. Этой версии достаточно для выполнения предлагаемых практических работ.

Основы имитационного моделирования

Теоретический материал учебника 10-го класса, посвященный **моделированию**, раскрывает суть основного метода познания информатики и применение системного подхода, широко используемых в других научных дисциплинах. В дальнейшем понятие *модели* используется постоянно — либо при описании того или иного способа организации автоматизированной обработ-

ки информации, либо как способ описания сложного объекта.

В любой вводной статье или лекции, посвященной моделированию в курсе информатики, всегда говорится, в частности, об огромном количестве практических задач прогнозирования и управления, которые решаются с помощью этого метода.

К сожалению, подтвердить эту абсолютно верную мысль на практических примерах в школьном курсе затруднительно — методы и средства, которые доступны обычно школьникам, позволяют решать только очень упрощенные или очень узкие задачи. Задачи же, сопряженные с реальной практикой, с помощью этих методов в школе решать трудно (например, численное решение оптимизационной задачи — предмет отдельного курса в высшем учебном заведении) — недостаточно времени и математической подготовки.

Приблизить материал к практическим задачам из реальной жизни можно, применив один из самых старых и мощных подходов к созданию компьютерных моделей — ***имитационное моделирование***.

В рамках темы **"Моделирование"** авторами рассматриваются не только традиционные вычислительные модели, но и модели имитационные: агентные, дискретно-событийные и системно-динамические.

Для практикума одна из ведущих мировых компаний — разработчиков средств имитационного моделирования, абсолютный лидер российского рынка, **компания The AnyLogic Company**, предоставляет специально адаптированную к условиям школьного курса версию среды AnyLogic, позволяющую создавать, демонстрировать и исследовать широкий спектр моделей из самых разных областей практической деятельности. Использование этой среды позволяет не только теоретически обсудить важность и возможности методов моделирования, но и продемонстрировать их важность и возможности для решения практических задач.

Для знакомства с теорией и практикой имитационного моделирования предлагается рассмотреть возможности среды на содержательных задачах, а также выполнить часть практикума по данной теме в виде двух практических работ.

Описание метода

Суть предлагаемого подхода: имитировать поведение системы, представив ее как множество простых взаимодействующих объектов, моделируя их поведение и взаимодействие и рассматривая результаты для всей системы в целом. Объект мы описываем как набор параметров и способов их изменения.

Применяя такой подход, мы можем:

- изучить поведение системы, не имея общих аналитических соотношений;
- проследить поведение системы в динамике, т.е. на каждом шаге процесса;
- исследовать поведение системы, в которой возможны случайные вариации поведения объектов.

Существует несколько видов таких моделей:

1. ***Дискретно-событийные модели.*** Такие модели описывают поведение системы как набор последовательных событий. Например, так можно описать рост пшеницы (процесс состоит из хорошо известных стадий) или работу учреждения (его работа описана инструкциями и регламентами). В такой системе описываются состояния (например, ожидание поступления на склад запасной части) и события, т.е. изменения состояния.

2. ***Агентные модели.*** Такие модели построены в виде набора взаимодействующих отдельных объектов (агентов), каждый из которых имеет какие-то заданные правила поведения и на основе поступающих сведений принимает решение об их применении. Например, такой агент может описывать поведение особи в стаде или человека в толпе. Поведение всей системы будет складываться как результат взаимодействия агентов, но спрогнозировать его математическим соотношением трудно. Например, поведение толпы на выходе из здания.

3. ***Модели системной динамики.*** Этот метод предназначен для исследования поведения сложных систем с большим количеством обратных связей и зависимых параметров — например, крупного города, большого производства, демографической ситуации. При таком виде моделирования систему представляют в виде взаимодействующих объектов. Каждый объект на самом деле может быть очень крупной системой, но при моделировании мы считаем его единым целым, пропускающим через себя, формирующим или поглощающим потоки разного рода (финансовые, материальные и другие).

Имитационные модели — один из самых распространенных методов анализа и предварительной проверки сложных систем, принятия решений в условиях множества разнородных связанных факторов, оптимизации расписания и т.д.

Строить такие модели можно различными способами. В особо сложных и специфических задачах (например, при моделировании транспортной системы города) такие модели разрабатываются как специализированные программы, но в целом ряде случаев можно разработать ее в рамках системы создания таких моделей.

Разберем несколько примеров таких моделей.

Первый пример — **моделирование потока пешеходов с помощью агентной модели**.

Предположим, что у нас есть какой-то путь пешеходов (подземный переход, магазин и т.д.). Нас интересует — каким образом повлияет установка витрины (или создание маленького мага-

зинчика на этом проходе) на движение пешеходов?

Определить эти параметры с помощью математической модели очень трудно, но можно описать среду движения, пешехода (и описать его движение) и создать имитационную модель, в которой понаблюдать за движением пешеходов.

Мы создаем модель, в которой пешеходы — это точки (точнее, круги), перемещающиеся по плоскости (см. *рис.* 1). Правила движения пешехода сравнительно просты — пешеход движется с постоянной скоростью (примерно от 0,5 до 1 метра в секунду) к заданной цели. Если он видит препятствие (стену) или медленно движущегося пешехода, он старается его обойти. Время в модели считается в условных единицах, для определенности будем считать их секундами.

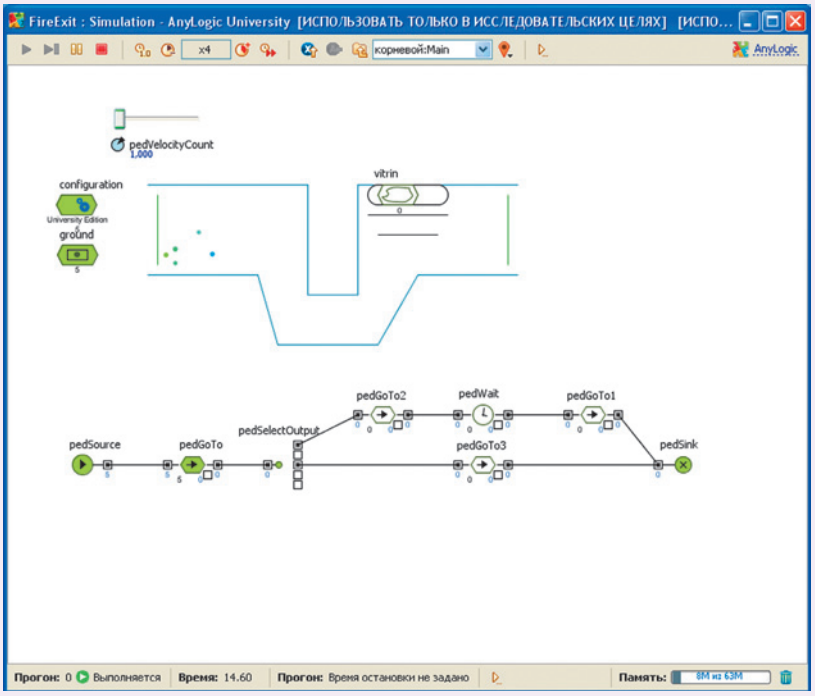


Рис. 1. Демонстрационное окно модели

Модель движения пешеходов мы создадим с помощью специализированной среды имитационного моделирования[1], в ней уже есть специально подготовленные средства (мы разберем их использование в практикуме) для моделирования пешеходов. Модель будет представлять собой программу, которая обеспечит учет перемещения пешеходов с заданной скоростью, сбор статистики и визуализацию происходящих событий.

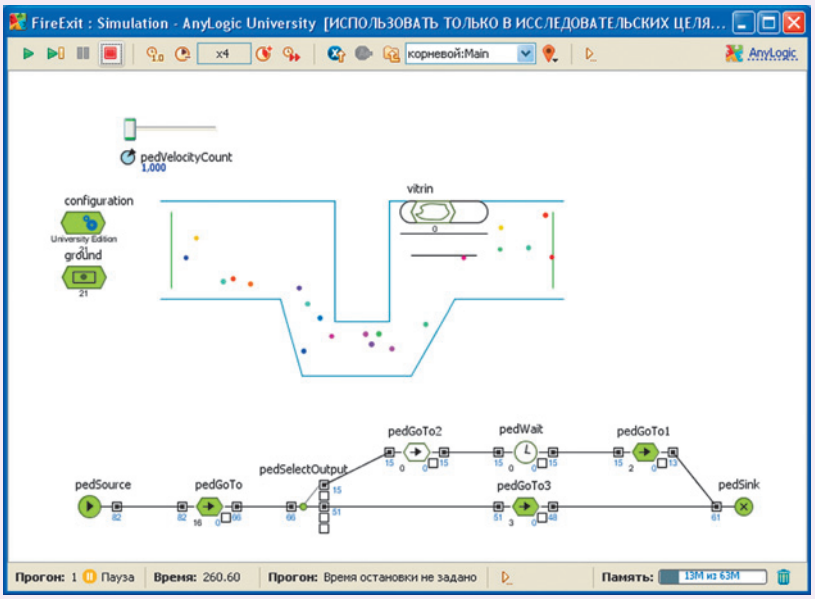


Рис. 2. Исследование модели: 1000 пешеходов в час

В окне модели видно:

1. Среду перехода. Пешеходы появляются в левой части и проходят направо с обычной скоростью до выхода. После поворота они видят с левой стороны прохода витрину и могут либо свернуть к ней, либо пройти на выход.

2. Алгоритм работы: пешеходы появляются (pedSource) с некоторой частотой — по умолчанию 1000 в час, идут (pedGoto) до некоторого места, где замечают витрину (отмечено чертой) и принимают решение (pedSelectOutput) идти дальше (большая часть, 7 из 10 пешеходов) или подойти к витрине (3 из 10). Итоговая цель пешеходов — выход (pedSink).

3. Параметр, которым мы можем управлять, — количество пешеходов в час.

Запустим модель и посмотрим результат. Для 1000 пешеходов в час никаких затруднений нет (см. *рис.* 2).

В переходе одновременно находится примерно 20 человек (подсчет статистики ведется на всех узлах).

Для 6000 пешеходов в час на повороте появляется затруднение, но в целом пропускной способности достаточно (см. *рис.* 3).

А теперь, при том же потоке пешеходов изменим модель — сделаем витрину более заметной, видимой почти от выхода. Для этого мы просто сдвинем место принятия решения о повороте (см. *рис.* 4).

Хорошо видно, что возникла пробка. Ее можно увидеть и численно: разница между вошедшими

[1] Использован пробный вариант среды имитационного моделирования AnyLogic v.6.5 University.

(576 на входе) и вышедшими (147 на выходе) стала гораздо больше, хотя к самой витрине народу подошло меньше (43 вместо 51).

Причина пробки — время, которое было потрачено частью людей на то, чтобы протолкнуться к витрине, а остальными — на то, чтобы все-таки пройти мимо. Хотя это время зависит от алгоритма обхода препятствия (и в другой версии системы моделирования может измениться), в реальности происходит то же самое.

Для создания более наглядной модели в качестве подложки можно использовать план здания, а стены-препятствия расставить точно так, как они стоят в реальности.

Конечно, создать такую модель для конкретных условий можно только тщательно исследовав поведение настоящих пешеходов: как они движутся, сколько из них смотрит на витрины, как изменяется при этом их движение. Именно поэтому такие параметры — предмет многочисленных и довольно сложных исследований.

Аналогичный подход можно применить к очень многим процессам, например, создав модели машин, веществ и аппаратов, исследовать организацию производства.

Такие модели — мощный и универсальный способ исследования влияния изменений в поведении отдельных объектов-агентов на всю модель. В математической модели, например, было бы очень сложно увидеть, как повлияет изменение тактики поведения жертв на популяцию хищников. Иногда очень простые изменения параметров приводят к существенно другим результатам, что в формулах можно учесть только после того, как эффект будет замечен.

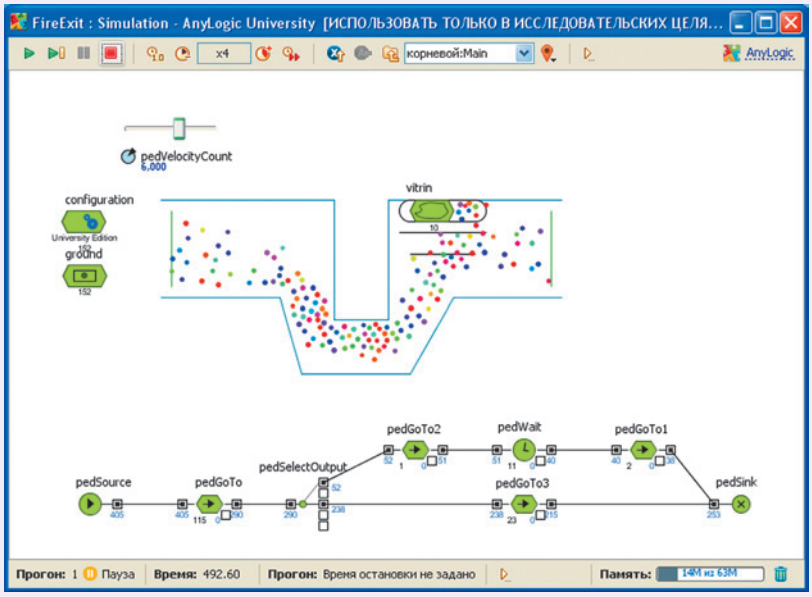


Рис. 3. Исследование модели: 6000 пешеходов в час

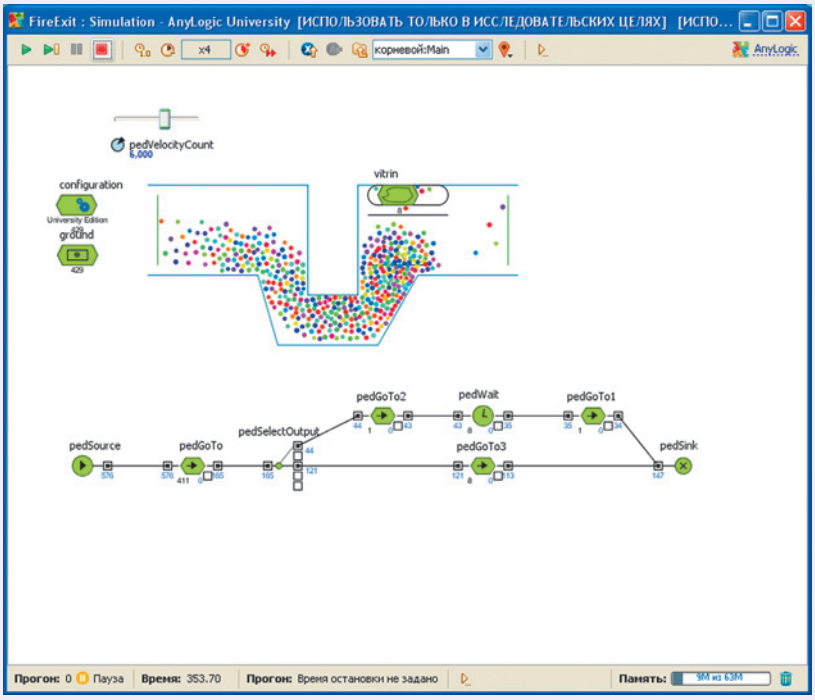


Рис. 4. Исследование модели: 6000 пешеходов в час, витрину замечают от поворота

Второй упомянутый нами вид моделей — **модели дискретно-событийные.**

В агентной модели агент (точнее — программный код, моделирующий его деятельность изменением параметров), как мы помним, принимал решения о своих действиях, исходя из восприятия окружения (то есть получая разрешенные ему сведения о состоянии среды). При моделировании же многих организованных процессов все решения уже приняты заранее, то есть известно, в каком порядке, кто и как должен действовать.

Моделируя такие системы, понятием агента (и самим агентным подходом) не пользуются, а используют понятие ***заявки на обслуживание*** — обращения к системе для достижения какого-то результата. Порядок обработки заявки задается в системе в виде некоторой последовательности событий: появления заявки, передачи заявки от одной стадии к другой, задержки (моделирующей выполнение стадии), накопления заявок в очереди.

При возникновении таких событий также оценивается достаточность ресурсов, например, занят или свободен какой-либо аппарат или комната.

Время в таких системах чаще всего представляется в виде условных дискретных шагов, поскольку заявка переходит из одной стадии в другую сразу, то есть дискретно.

Такой подход позволяет проанализировать работу системы под заданной нагрузкой, определить заранее "узкие" места, оценить количество необходимых ресурсов и качество работы системы в целом.

Рассмотрим такую модель на примере упрощенного **представления транспортного маршрута**.

Наш маршрут состоит из трех остановок: начальной, промежуточной и конечной. Маршрутное транспортное средство (будем его в дальнейшем называть маршруткой) имеет 12 мест и ходит по некоторому расписанию. В маршрутку можно посадить не более 12 человек.

На первой остановке люди только садятся, на промежуточной — выходят (если это предполагалось) и садятся (если есть места), а на конечной — только выходят.

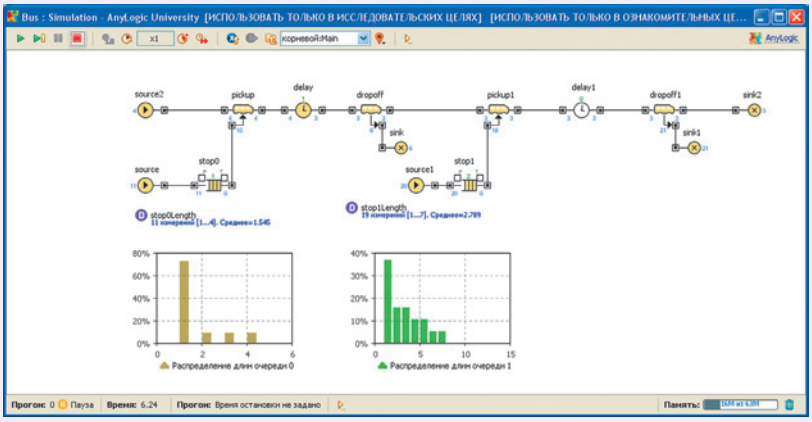


Рис. 5. Транзакционная модель маршрута

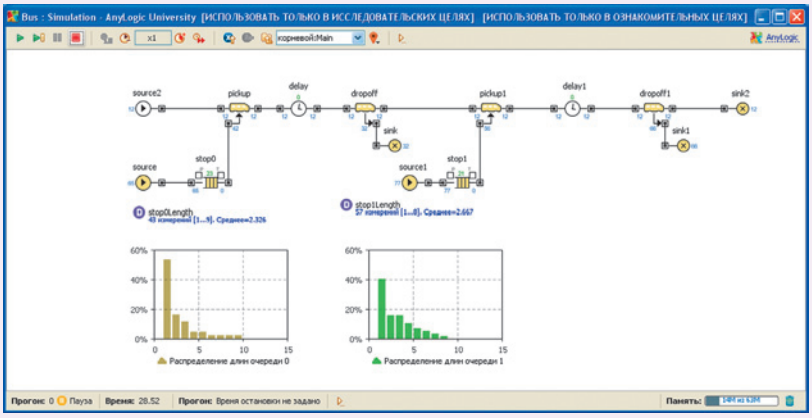


Рис. 6. Исследование модели

Для нас основным критерием работы линии будет количество ожидающих посадки людей на остановках. Еще один критерий, часто используемый в таких системах, — время, проведенное заявкой в системе.

Вот рабочее окно нашей модели — *рис.* 5.

В этой модели в отличие от предыдущей мы не используем анимацию для визуализации, поскольку нас интересуют только численные оценки. Длины очередей на каждом шаге мы фиксируем в специальных списках и отражаем на гистограммах, оценивающих длину двух очередей.

На *рис.* 5 видно, что длина очереди на первой остановке не превышала 4, а на второй, гораздо более интенсивной, доходит и до 7. Насколько такое допустимо — вопрос изначально заданных критериев.

Если пассажиры не будут учитывать расписания маршруток, то при длительном перерыве в их движении (в нашем случае — от 18-й до 30-й точки отсчета) ситуация может выглядеть так, как на *рис.* 6.

То есть на остановках начнут скапливаться ожидающие маршрутку (23 на первой остановке и 21 на второй, поскольку там последняя маршрутка прошла позже). Чтобы это скомпенсировать, потребуется больше, чем четыре маршрутки, потому что не обязательно выходят все люди и постоянно подходят новые. При необходимости можно подобрать более адекватное расписание или увеличить количество маршруток.

Эта модель упрощена, поэтому мы не оцениваем задержку при входе и выходе пассажиров[2], не рассматриваем работу станций-пересадок, возврат маршрутки назад, реальное время маршруток в пути. Очевидно, что при более тщательном подходе можно достаточно точно описать работу транспортной сети города и, оценив интенсивность появления пассажиров на остановках, спрогнозировать ее поведение.

Таким образом можно моделировать работу учреждений, складов, транспортных сетей, систем обслуживания, конвейерных производств, оценивать достаточность ресурсов и скорость их работы. В отличие от агентного подхода этот способ требует существенно меньших ресурсов, что позволяет строить и изучать достаточно сложные модели.

В целом ряде случаев применение обоих видов моделей — хорошая альтернатива реальным "пробам". Например, значительно целесообразнее проверять достаточность пожарных водоемов, помп и расчетов на

[2] В ряде городов введены в действие системы автоматического контроля пассажиров. В результате время посадки 15 человек может достигать 10–12 минут, что, конечно, стоит учесть в модели. Аналогично будет действовать и условие "оплата водителю при входе/выходе".

сетевой транспортной модели, а не в условиях реальных пожаров. Хотя в последние несколько лет решения, как кажется авторам, принимаются именно на основе практического опыта и внесистемных мотивов. Реальные потери, конечно, значительно убедительнее.

Особый вид имитационных моделей — **системной динамики**, разработан и используется для прогнозирования развития крупных систем, в которых поведение отдельных объектов взаимозависимо, причем связей много. Чаще всего такие системы рассматриваются в финансово-экономической сфере и в производстве.

Существенная особенность таких моделей — что в них мы рассматриваем не модели объектов, а взаимосвязь параметров.

В моделях системной динамики всю систему представляют в виде набора взаимосвязанных объектов следующих видов:

Название объекта	Назначение
Уровень	Переменная, накапливающая изменения
Связь	Обозначение влияния переменных
Связь с задержкой	Влияние с запаздыванием
Поток	Переменная, изменяющая значения уровней
Поток с регулятором	Изменение значения уровней с регуляцией. Изменение значения потока на основании вспомогательной переменной
Вспомогательные переменные	Переменные-параметры, определяющие поведение, но сами ни на что не влияющие
Константы	Неизменные (во время симуляции) значения
“Облако” (“Озеро”)	Неисчерпаемый источник или поглотитель

В качестве примера мы попробуем **исследовать работу таможни в некоторой стране**. *Рассмотрим типичную для таможни ситуацию, когда часть товара поставляется в обход правил уплаты сборов и контролирующие органы страны полагают, что бюджет недополучает эти средства.*

Предлагается идея: вместо того чтобы усиливать борьбу с контрабандой (это долго, дорого и не гарантирует успеха), государство увеличивает налог на товар, компенсируя свои потери.

Поставщики товара поднимают цену при продаже, чтобы скомпенсировать выросший налог.

Известно, что в таких случаях контрабанда возрастает, потому что становится выгодной большему количеству поставщиков и потребителей.

Возникает вопрос: сумеет ли государство скомпенсировать потери таким образом?

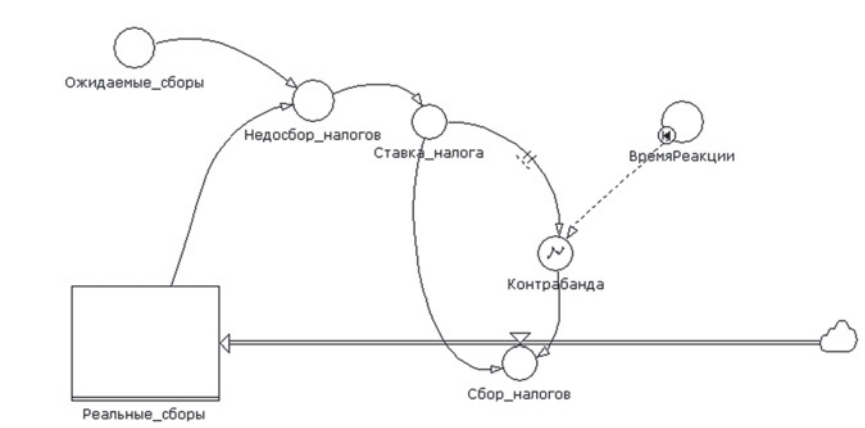
Составим модель: есть поток товара (который мы укажем в денежном эквиваленте — USD, Условных Системных Деньгах), с которого по заданной ставке взимается налог. Налоги определяют уровень реальных сборов.

Зная объем ввоза (предположим, он не меняется), мы предполагаем уровень сборов и можем оценить недосбор налогов. Исходя из недосбора, мы вычислим новую ставку налога.

Изменяя ставку, мы повлияем и на сбор налогов, и на уровень контрабанды. Уровень контрабанды изменится не сразу, а через некоторое время, требуемое для реакции системы.

Составим модель[3] (*рис.* 7).

“Проиграв” эту модель для 100 условных лет, мы выясним удивительную вещь — реальные сборы налогов не выросли, а упали. Можно экспериментировать с коэффициентами и усложнять



Год	Контрабанда	Недосбор_налогов	Реальные_сборы	Ставка_налога
0	USD1,10	USD0,00	USD80,00	USD0,00
20	USD2,19	USD5,39	USD74,61	USD4,31
40	USD2,44	USD6,07	USD73,93	USD4,86
60	USD2,49	USD6,21	USD73,79	USD4,97
80	USD2,50	USD6,24	USD73,76	USD4,99
100	USD2,50	USD6,25	USD73,75	USD5,00

Рис. 7

[3] Модель подготовлена в демонстрационной версии PowerSim Studio 8.

модель, но суть дела не изменится: простой рост налогов при нежелании или неспособности бороться с контрабандой только ухудшит ситуацию.

Такие сравнительно простые схемы встречаются достаточно часто и имеют свои названия. Приведенная в примере схема называется “Решение, обреченное на неудачу”.

Модель можно усложнить, введя уровень борьбы с контрабандой, и учесть в нем влияние на деятельность контролирующих органов возрастания объема сборов, влияние такой борьбы на время и стоимость транспортных перевозок, другие параметры.

Подобные схемы (иногда большие и сложные) позволяют изучать взаимное влияние происходящих социальных процессов и используют для принятия управленческих решений.

Системно-динамическое моделирование применяется для исследования и предсказания поведения сложных систем (например, социальных отношений) и активно используется для принятия решений.

ПРАКТИКУМ

Чтобы практически освоить основные виды имитационного моделирования и познакомиться с возможностями этого метода, мы рассмотрим построение таких моделей с использованием среды AnyLogic. Компания — производитель этой среды — российская компания The AnyLogic Company предоставляет для ознакомления пробную версию (*http://www.anylogic.ru/downloads*), которой нам более чем достаточно. Перед началом работы рекомендуем установить ее на своем компьютере.

Среда эта объектно ориентированная и для создания моделей опирается на язык Java. Особенности языка нам изучать не потребуется, для работы достаточно общих представлений об объектно ориентированном подходе и синтаксисе языка С.

Перед началом работы кратко опишем интерфейс среды. Основное рабочее окно среды выглядит примерно так — см. *рис.* 8.

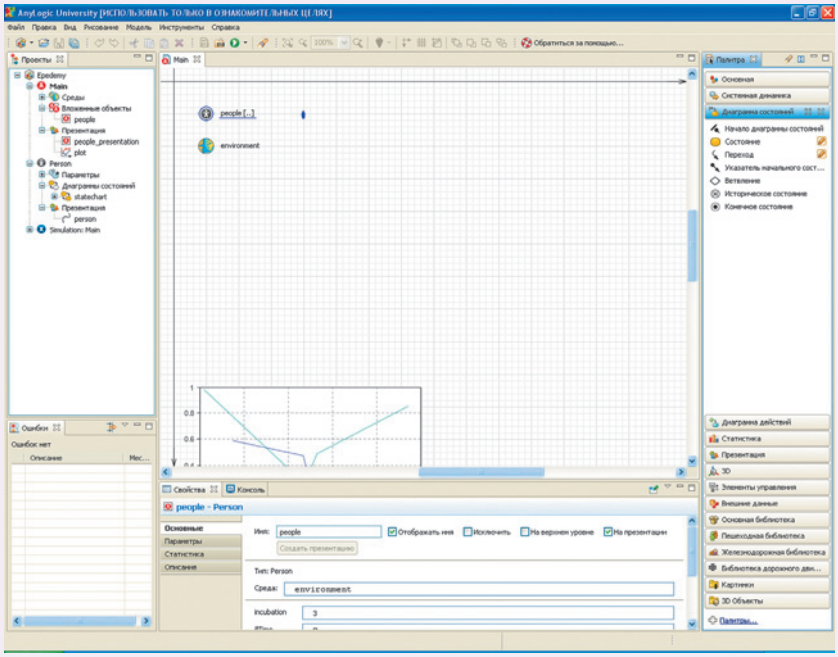


Рис. 8

С левой стороны вверху — дерево, в котором показаны открытые сейчас модели (на иллюстрации — только одна, Epedemy), а в моделях — объекты, из которых она состоит. Объекты размещены на двумерной плоскости — рабочем пространстве. В открытой сейчас модели мы видим объект-пространство — Main, объект-агент Person и стартовую страницу модели, на которой можно будет размещать настройки перед прогонами, — Simulation.

Объекты размещаются в рабочем пространстве — как в графическом редакторе. Объекты можно добавлять, перетаскивая их из палитр в правой части (сейчас открыта палитра “Диаграмма состояний”).

Объекты в модели имеют свойства и методы, которые можно описывать, задавая им значения в нижней центральной части рабочего окна. Наша задача с точки зрения среды делится на две фазы: подготовка модели (описание объектов и их взаимодействия) и “прогон” — запуск модели.

Начнем с наиболее иллюстративного вида имитационных моделей — агентных моделей, на котором и покажем, как именно это делается.

Практическая работа № 1
“Изучение движения учащихся через турникеты с помощью агентной модели”

Задача

В школе № 0 планируется повысить уровень обеспечения безопасности.

В первую очередь специалисты рекомендовали ограничить проход в школу посторонних, для чего на входе установят два турникета и начнут проверять всех входящих с помощью специальной карты.

По мнению директора школы, реальное применение такого подхода приведет к существенным затруднениям перед началом занятий.

Как подтвердить или проверить эти предположения?

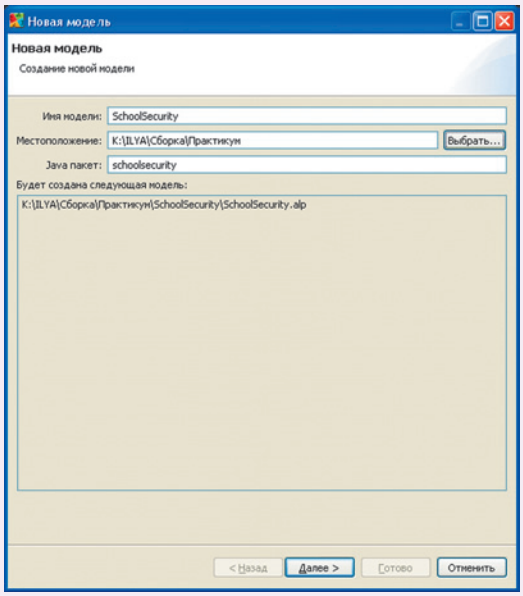


Рис. 9

Решение

Создадим модель, показывающую первый этаж школы с турникетами и агентами-пешеходами, единственной задачей которых будет — пройти через турникет.

При этом мы учтем, что проход через турникет (а точнее — проверка) занимает некоторое время, и время от времени будет попадаться человек без документов (например, без карточки), который будет возвращаться назад.

Время прохода может меняться (человек проверяет документы медленнее, чем автомат — карточки), вероятность появления "неправильного" документа — тоже.

Чтобы модель выглядела реалистично, найдем и загрузим типовой план первого этажа школы. Авторы использовали план из типового проекта № 45-621/1.

Запустим среду и командой "Файл\Создать" создадим пустую модель (см. *рис.* 9).

На втором шаге укажем, что модель создаем "с нуля", не используя заготовки.

Возьмем за основу типовой проект школы № 65-426/1. Найдем в сети план первого этажа, очистим его от лишних артефактов и разместим на рабочем листе: из набора компонентов "Презентация" перетащим на рабочее поле объект "Изображение" и добавим в его параметрах картинку плана этажа (см. *рис.* 10).

Вставленную картинку "растянем".

Теперь прямо поверх плана мы отметим стены (поскольку на картинке автоматически система стены выделить не может). Для этого построим ломаные вдоль стен. Обратите, пожалуйста, внимание — чтобы начать рисование ломаной, нужно дважды щелкнуть по значку "карандаш" справа от объекта "Ломаная" в наборе компонентов "Презентация".

Чтобы сделать нашу ломаную "стенами", нам нужно объединить все элементы в группу — выберите их, щелкнув на них мышью с прижатой клавишей Shift, нажмите правую клавишу мыши и выберите в меню "Группа/Создать". Появившуюся группу назовем Walls.

Создадим стартовую линию — "источник" пешеходов. Линию мы расположим перед крыльцом — с точки зрения модели нам не важно, откуда люди приходят к школе.

Чтобы ее нарисовать, перетащим объект-линию, подгоним размеры и зададим линии имя: enter.

Внутри школы отметим пару линий — выходов. На самом деле это, конечно, не выходы, а просто подъемы на 2-й этаж, но опять же — с точки зрения модели важно только то, что люди уходят с этажа. Схема, конечно, очень упрощена — в обычной ситуации людям надо еще, как минимум, раздеться.

Линии назовем exit1 и exit2, примерно так, как показано на *рис.* 11 (выделена линия exit1 — в начале и конце ее видны кружки).

Пока что на нашей модели ничего происходить не будет, поскольку мы ничего задающего движение пешеходов еще не сделали.

Построим общую логическую схему движения из готовых блоков. Блоки нужно перетащить по одному из раздела "Пешеходная библиотека" и расположить при-

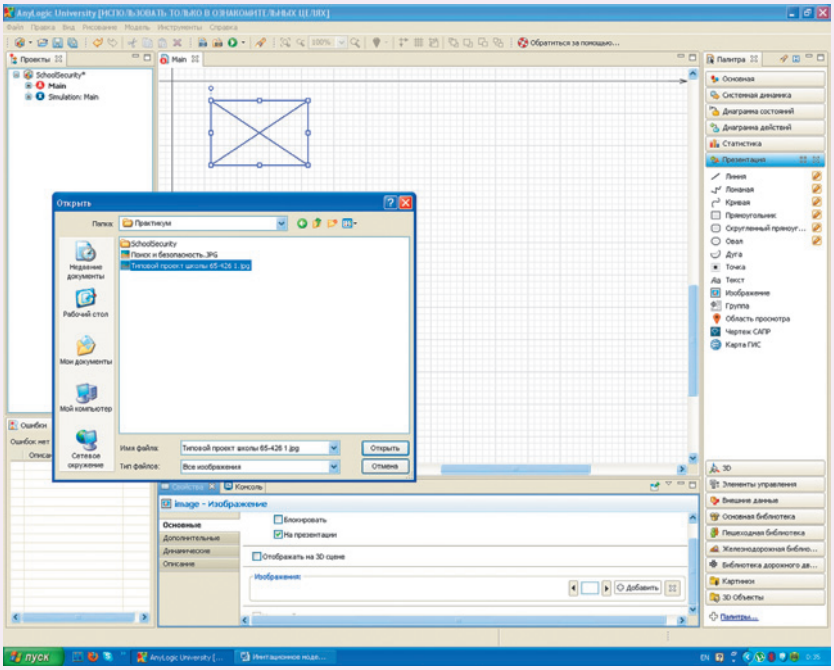


Рис. 10

мерно так, как показано на рисунке.

Блоки будут такие:

pedSource — источник пешеходов;

pedSelectOutput — выбор выхода;

pedGoto — движение пешехода.

Этих блоков будет два — поскольку у нас два выхода.

Сейчас для нас не так важно, что происходит за пределами турникетов, поэтому мы не учитываем того, что люди могут от каждого входа пойти к любому выходу.

Блоки нужно связать между собой. Для этого щелкнем два раза в прямоугольнике — порту для соединителей (см. *рис.* 12).

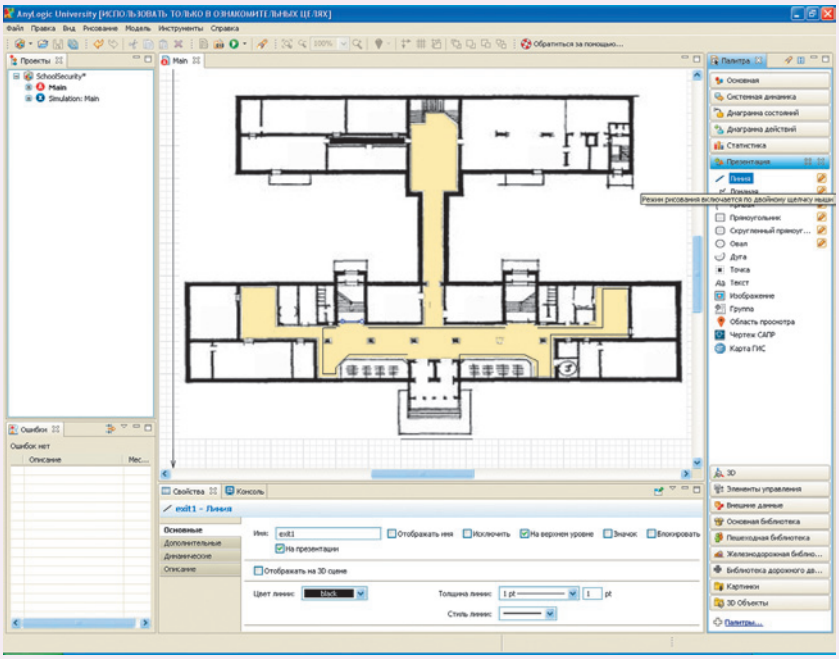


Рис. 11

Кроме них, нам на модели понадобятся еще два объекта без соединений:

pedConfiguration — конфигурация;

pedGround — объект-этаж.

В параметрах этажа нам нужно указать:

• имя этажа. Для этого в поле “Имя” напишем “ground1”;

• объект-стены. В разделе свойств нужно в поле “Стены” вписать название группы линий со стенами: Walls;

• зададим параметры источника пешеходов: ограничим количество пешеходов. Для этого поставим галочку “ограничено количество прибытий” и укажем в поле “Максимальное количество прибытий” значение 600;

• укажем объект-этаж. В поле “Этаж” напишем “ground1”;

• место появления. В соответствующем поле напишем имя линии — enter;

• укажем частоту появления новых пешеходов, исходя из следующих параметров: в школе учится 600 человек, и попасть внутрь они должны за полчаса. То есть параметр “Интенсивность” должен иметь значение 1200 человек в час.

Укажем параметры “разветвления” pedSelectOutput — коэффициенты предпочтения. Указать нужно вероятность выбора каждого направления, и в нашем случае они равны — по 0,5 первому и пятому исходу (остальные мы не использовали).

В каждом элементе движения pedGoto нужно указать цель — то есть линию выхода exit1 и exit2.

Запустим модель и, не меняя скорости, посмотрим, сколько времени займет проход пешеходов.

Стоит заметить, что время в модели — “виртуальное”. Фактически одна единица времени — это одна минута. То есть нам не нужно наблюдать за прогоном полчаса, достаточно 30 секунд.

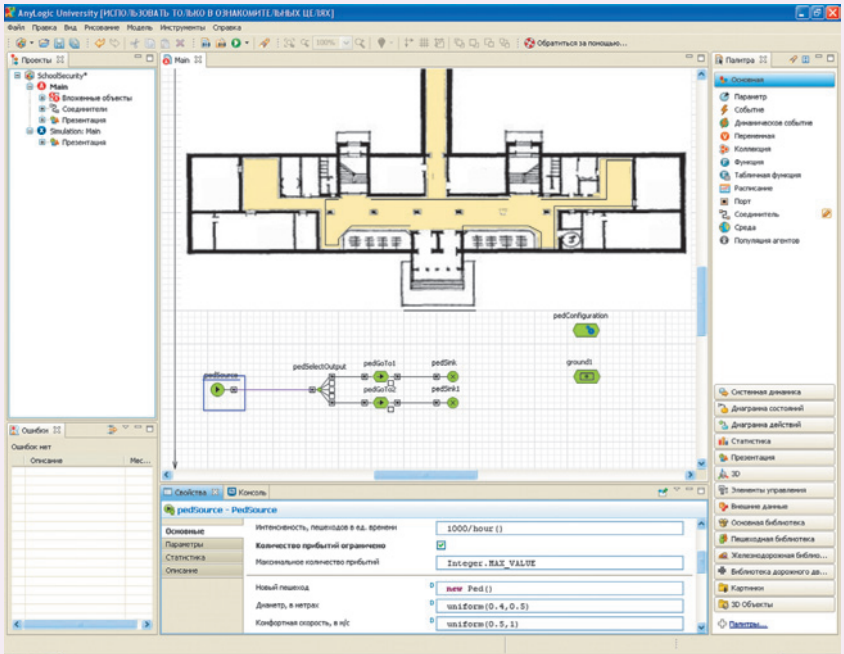


Рис. 12

Во время прогона мы выясним несколько мелких огрехов:

• пешеходы проходят через стенку между выходами и колоннами, поскольку мы их не учли в границах. Дорисуем недостающие линии и добавим их в группу;

• в некоторых случаях пешеход “не видит” свободного прохода к своему выходу, и это вызовет ошибку. Сдвинем линии выходов чуть внутрь.

В нашей модели скорость движения пешеходов различается, поэтому для уверенности мы сделаем несколько прогонов (*рис.* 13).

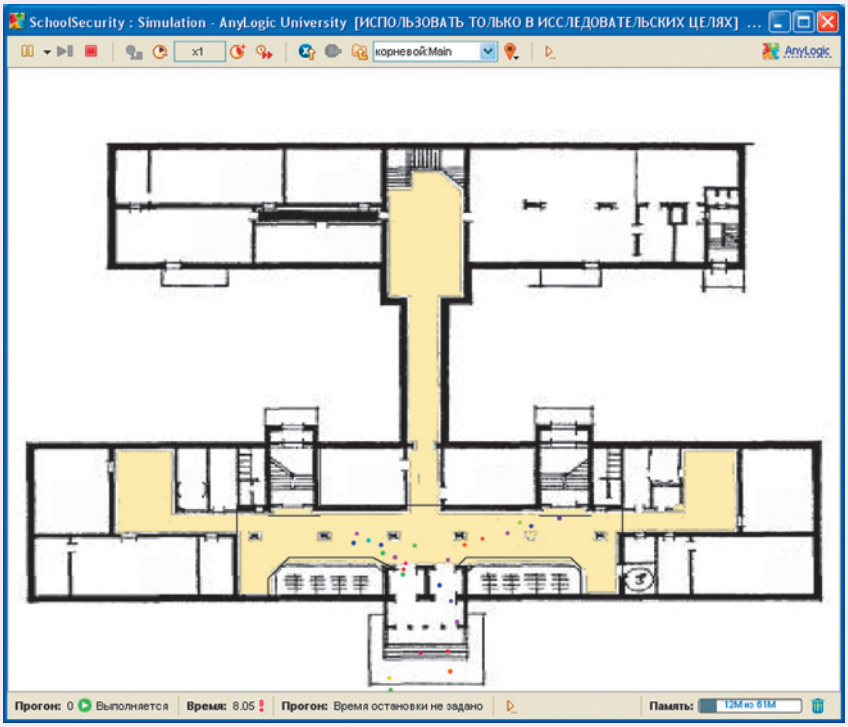


Рис. 13

Результат будет вполне предсказуемым — никаких препятствий нет, 600 человек попадают внутрь примерно за 30 минут (то есть по дороге нет затруднений — время зависит только от частоты появления и скорости движения).

Дополним нашу модель. Теперь мы поставим два турникета (нарисуем линии gate1 и gate2), обозначим двумя прямоугольниками зоны ожидания обработки и предусмотрим две линии — как цели для движения в случае отказа (например, забытой или отказавшей карты).

Время ожидания изменяется от 0,5 до 1 сек.

Для обработки ожидания нужно добавить два объекта-зоны: pedArea.

В схеме мы разобьем путь на три этапа: до турникета, ожидание пропуска и проход либо на этаж, либо на выход. Все объекты нужно связать, заполнив параметры.

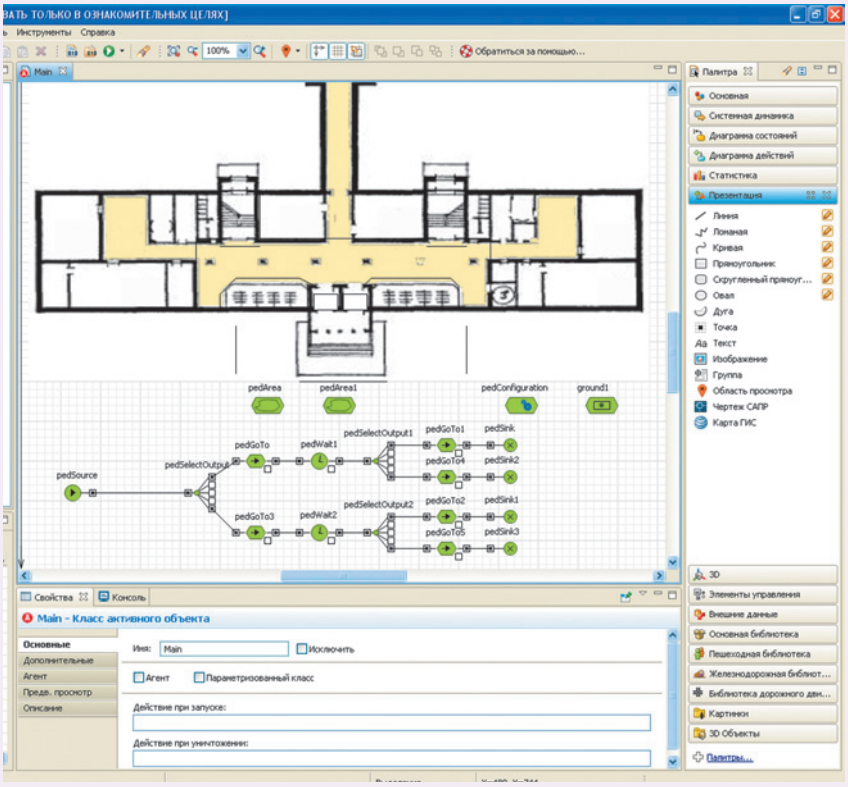


Рис. 14

В новых ветвлениях мы уже укажем неравное разделение: будем считать, что 95% посетителей имеют действительные карты (первый исход имеет коэффициент выбора 0,95), а 5% (0,05) — нет (см. *рис.* 14).

Снова запустим модель и обнаружим, что время выросло, но не очень существенно: до 32 минут.

Усовершенствуем нашу модель, чтобы больше не оценивать время “на глаз”. Для этого мы воспользуемся средствами сбора статистики и возможностью модифицировать класс “Пешеход”.

Мы видели, что во время работы модели некоторые пешеходы могут “заблудиться” и застрять на этаже. На реальный результат они повлиять не могут, поэтому мы можем их отбросить. Параметр, который нас на самом деле интересует, — это время, которое потребуется пешеходу, чтобы пройти через этаж. Изначально время входа и выхода у пешехода не фиксируется, поэтому создадим нового пешехода, отмечающего время.

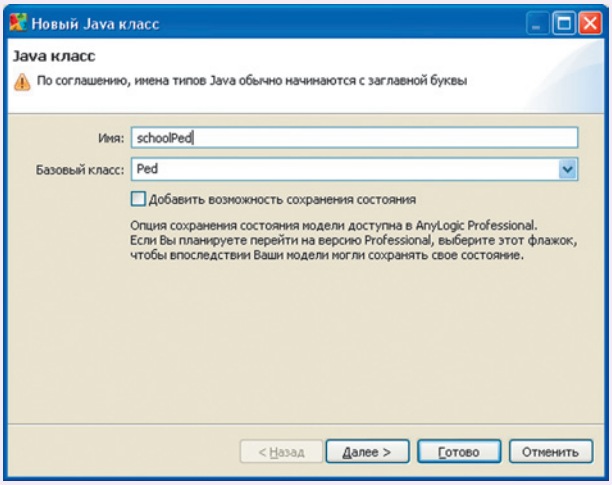


Рис. 15

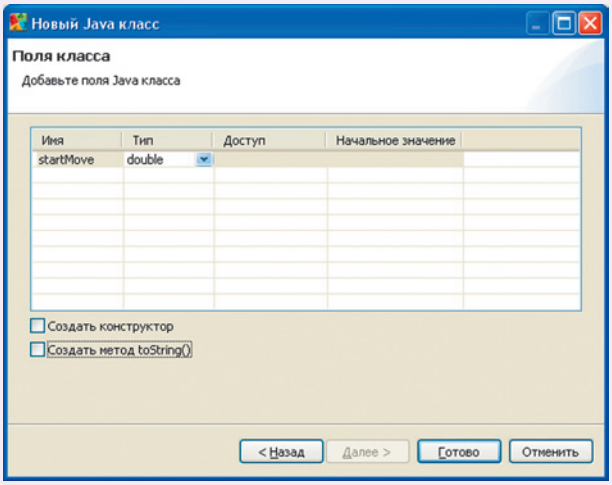


Рис. 16

Для этого:

• на названии модели нажмем правую кнопку, в меню выберем пункт “Создать\Новый Java-класс”;

• в появившемся окне укажем имя класса schoolPed и базовый класс — Ped (см. *рис.* 15);

• в следующем окне добавим параметр startMove — там будет храниться время входа. Тип этого параметра — double, поскольку именно так обрабатывается модельное время (см. *рис.* 16);

• галочки “Создать конструктор” и “Создать метод toString” уберем — мы будем записывать время сами;

• теперь укажем на объекте-источнике класс пешехода — schoolPed (вместо просто Ped, который был по умолчанию). Без этого мы не сумеем обратиться к новому параметру (в базовом классе его нет);

• в пункте “Новый пешеход” вместо new Ped() укажем new schoolPed();

• в пункте “Действие при выходе” запишем в добавленный нами параметр время: ((schoolPed)ped).startMove = time();

• поскольку в случае отказа пешеходы у нас идут в одно и то же место, модифицируем схему — исключим лишние объекты;

• добавим объект сбора статистики: из палитры “Статистика” перетащим объект “Статистика” и назовем его moveTime;

• сменим класс “пешеход” на всех объектах pedSink (так же, как на pedSource);

• на всех объектах-выходах укажем действие при входе, добавляющее данные к статистике: moveTime.add(time()-((schoolPed)ped).startMove).

Как видно, мы добавляем к статистике время — вычислив его как разницу между текущим временем и временем входа.

Теперь мы можем при выполнении прогонов не наблюдать за счетчиком, а просто отслеживать накопленные данные: минимальное, максимальное и среднее время нахождения прохода школьников.

Увидеть результаты можно просто щелкнув мышью на статистике (*рис.* 17–18).

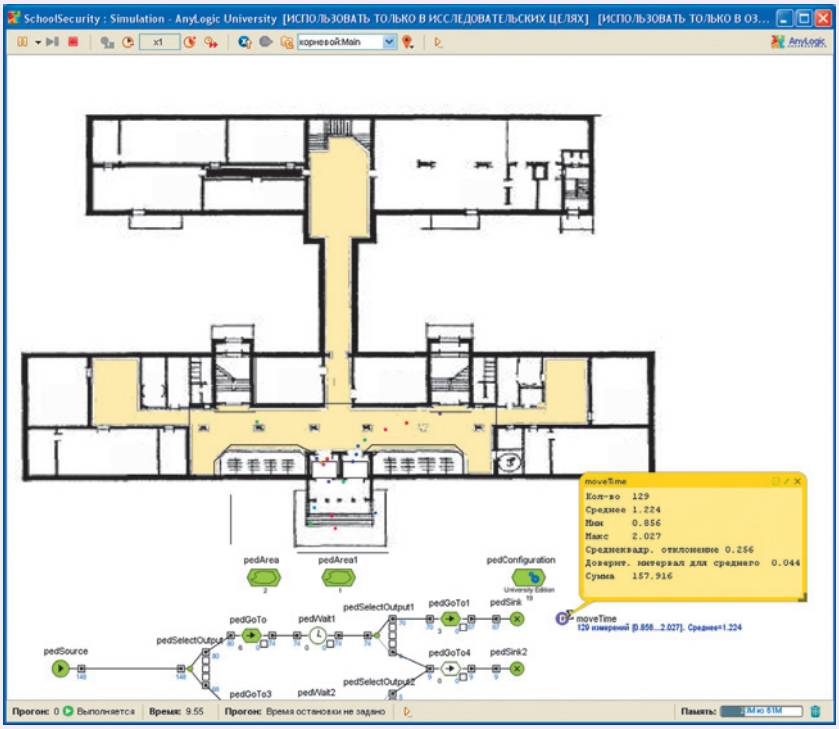


Рис. 17

Задания

1. Как изменится время, если карта будет обрабатываться от 1 до 2 секунд?

2. Сколько нужно будет времени на проход, если детям придется еще и переодеваться (задержка от 2 до 5 минут в раздевалке)?

3. Модель предполагает, что люди приходят в течение всех 30 минут равномерно. Тем не менее очевидно, что на последних 10 минутах плотность будет значительно выше. Исследуйте поведение модели с повышенной частотой появления людей.

Практическая работа № 2 “Дискретно-событийная модель работы учреждения”

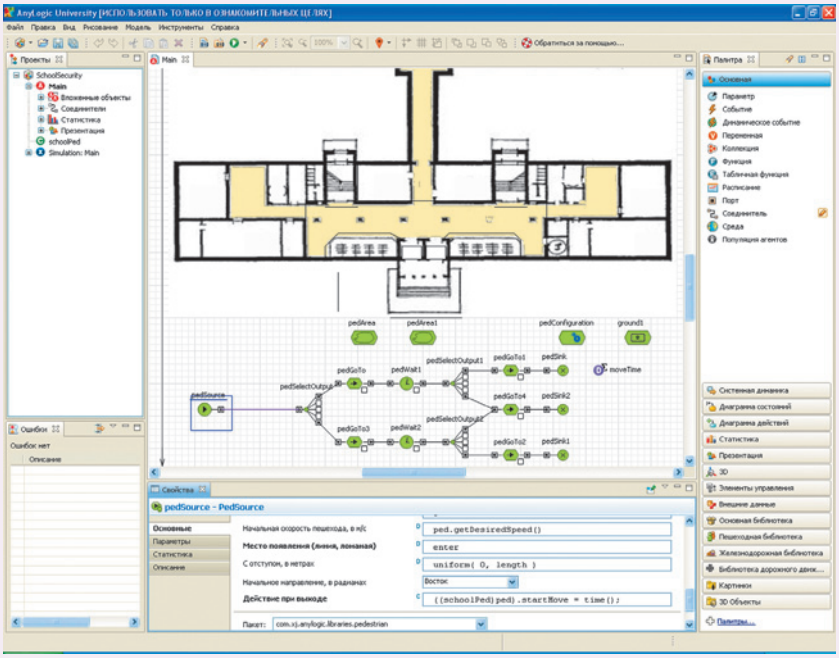


Рис. 18

Задача

Ежегодно работникам многих предприятий полагается проходить так называемый “профессиональный медицинский осмотр”. Процедура его построена примерно так: в назначенный день работник является в поликлинику, получает в регистратуре документы, получает от врача-терапевта направление и должен пройти осмотр у нескольких врачей-специалистов.

Люди выбирают специалиста сами, исходя из того, у кого очередь короче.

Общее время на проведение осмотра, выделенное лечебно-профилактическим учреждением, — 6 часов. Нам известно, сколько примерно тратят на осмотр и выдачу документов.

Требуется понять — сколько людей успеют пройти осмотр, не будут ли они где-то скапливаться при имеющихся нормах времени? Сколько в среднем проведет человек на осмотре, сколько максимально?

В модели, с помощью которой мы можем оценить эти параметры, нам не важна траектория движения людей, их взаимодействие между собой и т.д. Нам важно только время на каждом этапе обслуживания. При этом обслуживание для нас дискретно: нельзя же одновременно осматривать полтора человека.

Воспользуемся для моделирования специальным видом моделей, который называется дискретно-событийными моделями. Основная единица в нем — заявка на обслуживание, которая будет некоторое время обрабатываться на каждом этапе.

Создадим чистую модель (Файл/Создать/Модель), на втором этапе выберем “Чистый холст”.

Для создания модели нам понадобится, как и в предыдущих случаях, разместить необходимые объекты-этапы на рабочем поле и связать их между собой. Для дискретно-событийной модели мы воспользуемся объектами с закладки “Основная библиотека”.

Обработка начнется с создания заявки. Для этого мы перенесем из основной библиотеки объект “Source”.

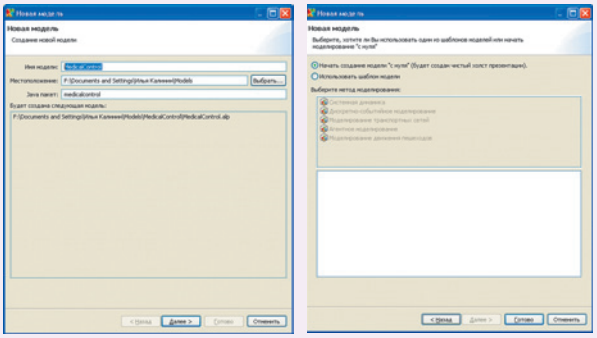


Рис. 19

Каждый этап будет состоять из двух объектов: “Очередь” (Queue) и “Ожидание” (Delay). Чтобы учесть выбор очереди к специалисту, нам придется вывести заявку из потока (с помощью объекта “Выход” (exit), и потом для каждого специалиста принять заявку в объекте Enter). После обработки заявки уничтожаются в объекте Sink.

Там, где обработка идет линейно, свяжем объекты, как и в предыдущих моделях.

Для трех специалистов получится примерно такая схема (*рис*. 20).

Как и в предыдущем примере, нам понадобятся объект для сбора статистики и гистограмма для отображения данных. Основным нашим параметром, статистику которого мы будем собирать специально, будет время, проведенное заявкой в системе, — то есть собственно время осмотра. Количество людей мы сможем узнать из чисел на объектах.

Обратите внимание: мы не связали очереди специалистов с основным потоком — заявки у нас будут попадать туда с помощью специальной функции.

Для работы модели нам понадобится создать новый класс заявок, как в прошлом примере для пешеходов. Назовем его Patient. Класс создаем на основе класса Entity (заявка) и описываем четыре дополнительных переменных:

```
boolean stomatolog;
boolean okulist;
boolean blood;
double ArrivalTime;
```

В них мы учтем специалистов, которых человек прошел, и время входа в здание. В конструкторе класса мы напишем начальные значения:

```
public Patient(double ArrivalTime)
 {
   this.ArrivalTime = ArrivalTime;
        this.blood = false;
        this.okulist = false;
        this.stomatolog = false;
 }
```

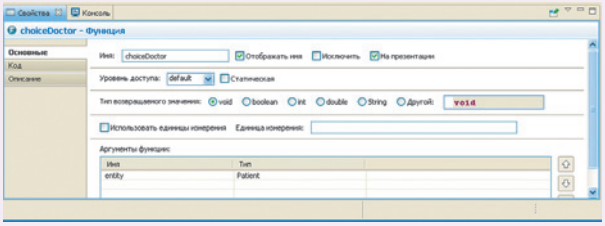


Рис. 21

Этот класс — Patient — мы должны будем указать на всех объектах, через которые проходит заявка.

Кроме этого, нам необходима будет специальная функция, которая будет выполнять выбор: в какую из очередей поставить заявку. Для этого из палитры “Основная” перетащим в рабочее окно компонент “Функция”, дадим ей название choiceDoctor и создадим один входной параметр: entity, то есть заявку из уже созданного нами класса Patient. Вот как примерно будет выглядеть страница основных свойств функции (*рис.* 21).

Конечно, функции необходим и программный код, который и будет ее реализацией. Код запишем в разделе “Код”. Реализовать его можно по-разному, автор предлагает такой вариант:

```
if                    ((entity.blood)&&(entity.
stomatolog)&&((entity.okulist)))
     enter3.take(entity);
else {
     int bloodSize = ((!entity.blood)?( blood.
size()):(255));
   int okulistSize = ((!entity.okulist)?
                      (okulist.size()):(255));
   int stomatologSize = ((!entity.stomatolog)?
                    (stomatolog.size()):(255));
     if ((bloodSize <= okulistSize)&&
                (bloodSize <= stomatologSize))
     {
          entity.blood = true;
```

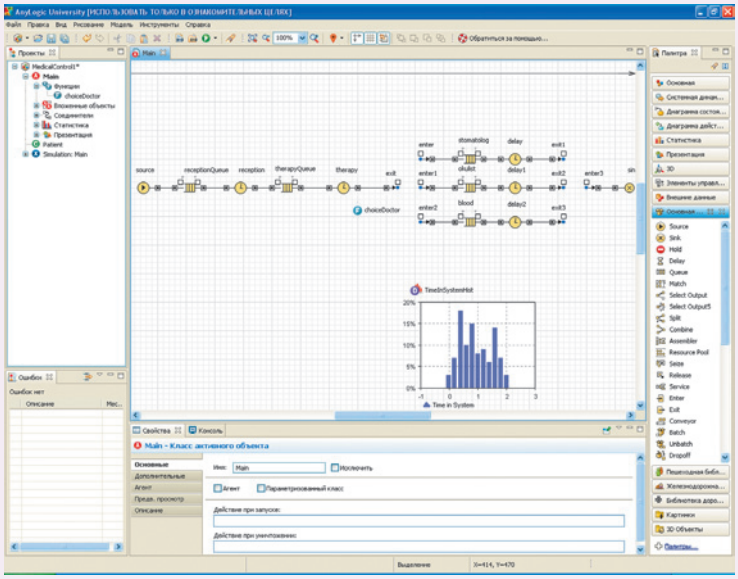


Рис. 20

```
          enter2.take(entity);
     } else
          if (okulistSize <= stomatologSize)
          {
               entity.okulist = true;
               enter1.take(entity);
          } else
          {
               entity.stomatolog = true;
               enter.take(entity);
          }
}
```

Логика работы этой функции такова:

1. Если человек прошел всех специалистов — он свободен.

2. Если нет, то мы выясняем размер очереди для каждого из них. Для тех специалистов, которые уже обработали заявку-обращение, делаем этот размер заведомо неприемлемым.

3. Выбираем самую короткую очередь, ставим туда заявку в нее и делаем отметку, что человек этого специалиста прошел — все равно, пока заявка не выйдет из очереди, эти данные не используются.

Этой функцией мы будем пользоваться всякий раз, когда нам нужно будет выяснить, куда теперь направить заявку. Для этого во всех объектах exit укажем на странице свойств, что при выходе нужно вызвать действие: choiceDoctor(entity).

Основная часть модели готова, но нужно задать еще несколько важных параметров: время задержки. Очевидно, что время задержки на разных этапах будет разным для разных заявок. Это может быть обусловлено массой причин, но сами причины нам не важны — важно, как распределится время обработки каждой заявки на каждом этапе. У нас нет полных статистических сведений, но есть средние оценки. Мы будем считать время обработки заявки случайной величиной. Форма ее распределения нам неизвестна, воспользуемся простейшим подхо-

дом — предположим, что это “треугольник” с вершиной в области среднего значения.

Предположим, что:

1. Время поиска и выдачи документов в регистратуре — от 3 до 10 минут, в среднем — 5. Зададим время задержки так: triangular(3, 5, 10). Для начала у нас только одно окно выдачи.

2. Терапевт работает быстрее, время осмотра одного человека — от 2 до 5 минут, в среднем — 3.

3. Стоматолог — от 5 до 15, в среднем — 10 минут.

4. Окулист — от 5 до 10, в среднем — 7.

5. Забор крови — от 1 до 5 минут, в среднем — 3.

Время на переходы мы игнорируем — предположим, что все происходит на одном этаже и человек видит все очереди.

Нетрудно посчитать, что формально максимум времени на полный осмотр — 45 минут. Теперь посмотрим, как это будет происходить в более точной модели.

Будем считать, что у нас прибывает один пациент в две минуты, то есть в параметрах источника укажем скорость прибытия 0,5 единицы. Там же проверим, что не забыли указать класс заявки при создании:

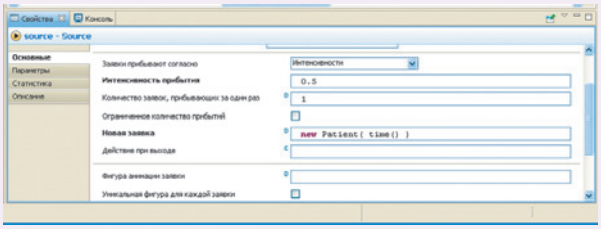


Рис. 22

Ограничим время работы модели: выберем ветку Simulation и в ее свойствах на закладке “Модельное время” укажем конечное время работы — 480 минут.

Теперь можем запустить модель. Сразу можно отметить несколько особенностей:

1) у нас появляется очередь в регистратуру, причем она все время растет;

2) к окулисту и стоматологу всегда стоит очередь, а на анализ крови ее нет.

Результаты исполнения таковы: примерно через 5,5 часа очередь в регистратуру переполняется. За ВСЕ это время осмотр смогли пройти только 30 человек, причем солидная часть из них потратила на это более трех часов (*рис*. 23).

Очевидно, что без снижения времени на осмотр и улучшения работы регистратуры такая система не сможет фактически исполнить свои задачи.

Задания

1. Почему на анализ крови практически не было очереди?

2. Как отразить в параметрах модели тот факт, что коридор, где ждут своей очереди на осмотр, — не безразмерный?

3. Что произойдет, если в очереди будут появляться заявки с длительным временем обслуживания — например, любители поговорить о здоровье? Какие параметры нужно будет изменить, чтобы это выяснить?

4. Что произойдет в системе, если за счет внедрения электронной карточки врачи смогут уменьшить время на заполнение документов на две минуты каждый?

5. Как повлияет на такую систему появление второго сотрудника в регистратуре? Вы можете реализовать это в модели с помощью объектов Service и ResourcePool (они заменят очередь и обслуживание в регистратуре) основной библиотеки. Пример использования — модель Bank из обучающих моделей AnyLogic.

Как видно даже из очень небольшого материала, метод имитационного моделирования позволяет решать огромный спектр практических задач. С методической точки зрения ценность его еще и в том, что он ярко демонстрирует саму суть понятия “моделирование”, не требуя высокой математической и технической подготовки.

Применение современной среды моделирования позволяет не только говорить о методе на занятиях, но и провести массу эффектных и полезных проектных работ, темы для которых можно найти в самых разных школьных предметах, собственном окружении, личной практике.

Литература

1. *Качала В.В.* Основы теории систем и системного анализа. М.: Горячая линия-Телеком, 2007.

2. *Форрестер Дж.* Основы кибернетики предприятия (индустриальная динамика). М.: Прогресс, 1971.

3. *Шеннон Р.* Имитационное моделирование систем — искусство и наука. М.: Мир, 1978.

4. *Медоуз Д.Х.* Азбука системного мышления. М.: Бином. Лаборатория знаний, 2010.

5. *Арнольд В.И.* Теория катастроф. 3-е изд., доп. М.: Наука. Гл. ред. физ.-мат. лит., 1990.

6. *Калинин И.А., Самылкина Н.Н.* Информатика. 10-й класс. Углубленный уровень. Учебник. М.: Бином, 2013.

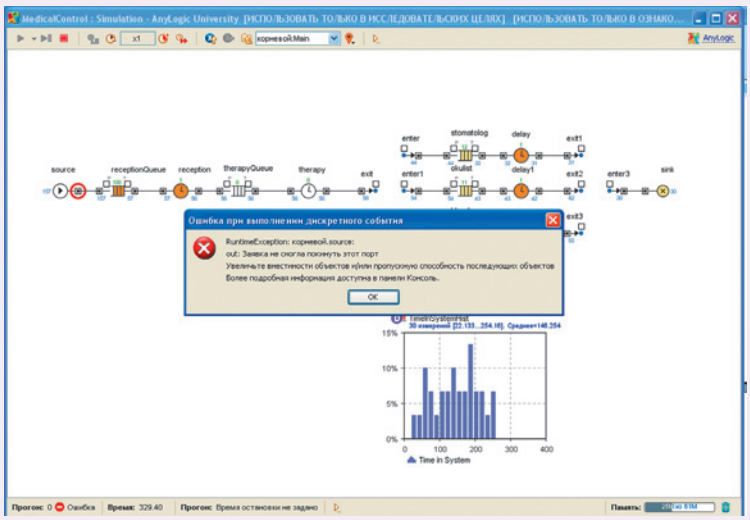


Рис. 23

Обработка текста

Технология компьютерной обработки текста

И.А. Калинин,
к. п. н., доцент кафедры информатики и прикладной математики МГПУ

Н.Н. Самылкина
к. п. н., доцент, профессор кафедры теории и методики обучения информатике МПГУ

Представление информации в виде текста остается до сих пор одним из наиболее универсальных и стало одним из первых доступных для обработки с помощью компьютера. Энциклопедический словарь дает такое определение понятию “текст”: “**Текст** — это упорядоченный набор слов, предназначенный для того, чтобы выразить некий смысл. В лингвистике термин используется в широком значении, включая в себя и устную речь”.

Представление информации в виде текста при обработке с помощью компьютера близко к этому определению. Под “текстовым” понимают такое представление информации, при котором она представлена в виде записи слов (логических элементов) некоторого языка и доступна для чтения человеком.

Язык для такого представления характеризуется в первую очередь некоторым алфавитом, т.е. допустимым набором символов. Поскольку компьютер работает только с двоичным кодом, то для записи и обработки текста требуется взаимно и однозначно сопоставить символы и некоторые двоичные коды. Такое *правило сопоставления кодов и символов алфавита называется* ***кодировкой текста***.

Для русского языка существует несколько вариантов кодирования. Во-первых, это кодировки на основе стандарта ASCII: CP866 (DOS), KOI-8R (UNIX), CP1251 (Windows) и MacCyr. Во-вторых, это общий универсальный стандарт кодирования UNICODE. Как правило, программные средства поддерживают либо один из стандартов первой группы, либо один из группы и UNICODE. Как вы знаете, при необходимости преобразовать текст из одной кодировки в другую не очень сложно.

С точки зрения хранения чаще всего для работы с текстом применяются ***текстовые файлы*** — самый простой, компактный и легкоиспользуемый с технической точки зрения способ хранения и передачи текста. ***Текстовый файл*** *представляет собой последовательность символов, разбитых на абзацы или строки.* Понятие “текстового файла” не предусматривает строго заданного формата или расширения. Тем

не менее, помимо характерной для той или иной ОС таблицы кодировки, в текстовых файлах могут применяться три основных способа деления текста на строки (абзацы):

1. Windows (DOS) — символы “Возврат каретки”+“Перевод строки” (CR+LF);
2. Unix — символ “Перевод строки” (LF);
3. MacOs — символ “Возврат каретки” (CR).

Текстовые файлы применяются для самых различных целей и часто оказываются формой хранения данных, описанных сложными формальными языками. Эти файлы часто используются для записи конфигурации ПО, документирования, переноса данных, описания HTML- или XML-кода.

Анализ текста на естественном языке

Традиционно, когда мы говорим об обработке текста, мы подразумеваем его создание и оформление — например, в виде макета печатного издания, со всеми его особенностями. Это важнейшее (и самое известное) применение возможностей компьютеров в этой области, но далеко не единственное. Не менее важны для нас самые разные средства, позволяющие автоматизировать и более интеллектуальные действия, в частности, извлечение из текста новой, более полезной информации, верификация и т.п. Пример таких действий — анализ текста.

До тех пор, пока текст невелик, мы можем проанализировать его без применения специальных средств, прочитав и обдумав. Хотя даже в этом случае многое будет зависеть от того, насколько хорошо развито умение извлекать из текста полезные сведения. В том случае, когда необходимо сделать контент-анализ достаточно объемной книги (нескольких книг) или множества любых текстовых источников, такой анализ становится как минимум затруднительным для пользователя — на чтение и обдумывание нужно потратить немало времени. Работа такого рода в настоящее время становится обязательной составляющей любой учебной или профессиональной деятельности. Когда времени мало, а текста много, возникает необходимость в автоматизированной обработке текстовых источников.

Анализ текста на естественном языке — очень многогранная задача. Различные методы и средства анализа разработаны для многих узких областей — это задачи поиска, перевода, выявления закономерностей и т.д. Мы продемонстрируем несколько специализированных методов автоматической обработки текста, которые в разных формах уже очень широко применяются в различных текстовых редакторах и текстовых процессорах.

Регулярные выражения. Одним из наиболее эффективных средств автоматической обработки текста являются *регулярные выражения*.

По сути, регулярное выражение — это шаблон, который можно сравнить с фрагментом текста. Регулярные выражения могут использоваться для поиска в тексте, выборки фрагментов и их преобразования. Таким способом можно найти и выделить:

1. Имена собственные — фамилии, имена, отчества людей, названия городов, рек и т.д.
2. Численные факты — объемы, количество чего-либо и т.д.
3. Данные, соответствующие шаблону, — телефоны, адреса и т.п.
4. Фразы, построенные определенным образом.

Регулярные выражения в разных видах реализованы в составе:

— текстовых редакторов и процессоров. Частичный механизм регулярных выражений реализован даже в Microsoft Word, более функциональный механизм есть в других текстовых редакторах, например, в Notepad++ или в PsPad;

— специализированных средств работы с регулярными выражениями. Как одно из удобных и свободных средств такого рода можно назвать Expresso, основанный на использовании механизма регулярных выражений среды .Net. Средства такого рода позволяют отслеживать исполнение выражения, результаты его применения к тексту, создавать библиотеки готовых шаблонов;

— программных библиотек, реализующих регулярные выражения в средах разработки, языках сценариев и др. Фактически подавляющее большинство современных систем либо имеют такие средства в своем составе (PERL, PHP, JavaScript), либо они могут быть без затруднений добавлены.

В специальных аналитических системах, предназначенных для автоматической обработки текста, кроме регулярных выражений используются специальные словари и заранее подготовленные описания логических конструкций. В нашем распоряжении таких инструментов нет, но ничто не мешает выполнить анализ самостоятельно, сконструировав нужные выражения.

Синтаксис регулярных выражений. Регулярное выражение составляется как описание последовательности символов. Символы, которые не относятся к специальным, считаются частью шаблона. В простейшем случае это просто строка, которую мы ищем. Например:

“Иван Грозный”

Такое выражение, что очевидно, позволит найти в тексте все вхождения подстроки “Иван Грозный”.

Помимо обычных символов, можно обозначить некоторые специальные. Для их записи перед ними ставится знак “\”:

Обозначение	Символ
\t	Табуляция
\r	Возврат каретки
\n	Новая строка
\a	Звонок
\\	Слэш
\.	Точка
^	Начало строки
$	Конец строки
\b	Граница. Первый или последний символ слова

Также действует правило экранирования: если нужно использовать символ, имеющий специальное значение, то перед ним ставится “\”.

Но постановка нашей задачи анализа текста такова, что мы, очевидно, не сможем задать все возможные подстроки для поиска имен. Кроме того, такая подстрока не подходит даже в том случае, когда между “Иван” и “Грозный” будет разрыв строки или два пробела.

Начнем с простого случая — обработки более сложного разрыва. Для этого мы опишем символы, которые могут стоять между словами. Для описания такого набора в регулярных выражениях предусмотрены *символьные классы*. Класс описывается набором символов в квадратных скобках. При записи можно либо перечислить подряд все символы класса, либо указать диапазон. Если сразу после открывающей скобки стоит символ “^”, то класс состоит из всех символов, НЕ перечисленных в скобках. В остальных местах этот символ означает начало строки. Символ “точка” в классе обозначает именно точку — а не любой символ. Некоторые классы уже описаны и можно использовать их сразу (см. таблицу внизу).

Конструкции вида `\p{}` относятся к использованию Unicode-кодировки. Они позволяют выделить символы с определенными свойствами во всех алфавитах. Для работы с русскими буквами предпочтителен этот способ, поскольку большинство реализаций регулярных выражений не включают русские буквы в обычные классы по соображениям совместимости.

Пример:

```
"Иван[\s]Грозный"
```

Такая конструкция подойдет для тех случаев, когда слова разделяются табуляцией или переводом строки, но символ-разделитель будет только один.

Для решения этой проблемы используются *квантификаторы* — указания на количество вхождений, которые обозначаются следующими символами (см. таблицу вверху).

Символ	Значение
`*`	Любое количество, включая 0
`?`	Может быть (не более одного вхождения)
`+`	Не менее одного вхождения подстроки
`{a}`	Ровно *a* вхождений
`{a,b}`	От *a* до *b* вхождений

В нашем случае конструкция будет такой:

```
Иван[\s]+Грозный
```

Но основная проблема не решена: мы не можем перечислить все имена. Остается перечислить признаки имени собственного и, исходя из них, сформировать регулярное выражение.

Имя собственное, как мы знаем, начинается с большой буквы и (в русском языке) не содержит цифр. Поэтому мы можем использовать для выделения имени собственного такую конструкцию:

```
\p{Lu}+\p{Ll}+
```

То есть обязательно в начале серии букв как минимум одна большая буква (сокращение в скобках от ***L**etter **U**pper case*), а потом обязательно хотя бы одна маленькая (сокращение в скобках от ***L**etter **L**ower case*). Первая же проверка покажет, что этот шаблон выделит и все слова в начале предложений. Кроме того, при использовании этой конструкции мы захватим и один лишний символ — разделитель.

Придется их исключить. Для этого мы можем использовать при выделении нужной нам части выражения группировку и предварительные условия — так называемые *высказывания*. Особенность условий в том, что они не становятся частью захваченной части текста. Группировка выполняется с помощью скобок, а чтобы указать, что группа — это предварительное условие и захватывать его не нужно, используют обозначения:

Обозначение	Значение
`?<!`	Не содержит предварительно выражения…
`?<`	Содержит предварительно выражение…
`?=`	После содержит…
`?!`	После не содержит…

Класс[1]	Обозначение	Описание
`[A-Za-zA-Яа-я0-9_]`	`[:alnum:]` или `\w` для PCRE[2]	Символы, образующие слово
`[^\w]`	`\W` для PCRE	не алфавитно-цифровой или ‘_’ символ
`[ \t\v\r\n\f]`	`[:space:]` или `\s` для PCRE	Пробельные символы
`[^\s]`	`\S` для PCRE	Непробельные символы
`[a-za-я]`	`\p{Ll}, [:lower:]`	Прописные буквы
`[A-ZA-Я]`	`\p{Lu}, [:upper:]`	Заглавные буквы
`[0-9]`	`\p{N}, [:digit:], \d` — для PCRE	Цифры
`[^\d]`	`\D` для PCRE	Все не-цифры
	`.` (символ “точка”)	Любой символ, кроме символа новой строки

[1] Классы описаны для русского и английского языков.

[2] PCRE (***P**erl **C**ompatible **R**egular **E**xpression*) — регулярные выражения, совместимые с языком Perl. Один из наиболее стандартизированных и развитых диалектов регулярных выражений.

В итоге получаем:

```
(?<![\.]\s+)(\p{Lu}+\p{Ll})
```

Конечно, этот способ небезупречен. Например, нельзя распознать идущие подряд имя и фамилию. Регулярные выражения позволяют нам рассматривать разные варианты, записываемые через символ "|". Обратите внимание: между словами с большой буквы должен быть либо пробел, либо тире:

```
(?<![\.]\s+)((\p{Lu}+\p{Ll}+[\s\-]+
\p{Lu}+\p{Ll}+)|(\p{Lu}+\p{Ll}+))
```

Анализ достаточно длинного и разнообразного текста покажет, что и этот вариант нуждается в улучшении. Во-первых, предложение может начинаться не только после точки, но и находиться внутри текста как цитата — в кавычках. Во-вторых, имена собственные бывают и более чем из двух частей. Перечислять варианты было бы затруднительно, поэтому мы снова воспользуемся группировкой и укажем, что группы повторяются:

```
(?<![\.^$\"]\s+)((\p{Lu}+\p{Ll}+[\s\-]{1})+)
```

Существенным плюсом регулярных выражений является то, что для их работы не требуется словарь, то есть мы выявим в тексте даже те факты (например, имена), о которых ничего не знали раньше. Конечно, для этого запись текста должна, по возможности, соответствовать правилам набора, например, правилам машинописного набора, правилам записи телефонов и адресов и т.п.

Частотный анализ текста. Регулярные выражения — мощный многоцелевой механизм обработки символьных строк. Область их применения очень велика, но для целого ряда задач они могут быть только вспомогательным средством.

Регулярные выражения сами по себе не могут применяться для анализа структуры текста, связи его с другими текстами, вынесения каких-либо суждений о тексте в целом. Для этого требуются более сложные методы анализа.

Под словом *анализ* мы будем понимать действия, направленные на выявление каких-либо существенных особенностей исследуемого объекта. При этом мы будем придерживаться общего плана:

1. Задание цели анализа, т.е. зачем мы выявляем особенности и параметры. Анализ никогда не выполняется "просто так". Больше того, результат анализа *всегда* зависит от целей.

2. Определения. Перед началом процедуры мы должны четко определить объект исследования, способ его представления, признаки, которые будем выявлять. Определить все это нам нужно в терминах предметной области — примерно так, как мы определяли понятия "слово" и "предложение" в терминах текста, т.е. набора букв.

1. Выбор метода анализа. Метод анализа подбирают исходя из целей и выделенных ресурсов (например, времени). Метод, как правило, имеет ограниченное применение, поэтому важно точно определить — какие методы мы можем использовать.

2. Планирование процедуры. Планирование включает в себя строгое описание наших действий, включая критерии достижения результата (или его недостижимости).

3. Подготовка материалов. Исходный объект редко готов к обработке сразу. Как правило, необходима предварительная подготовка — сбор, очистка, преобразование данных к виду, предусмотренному методом.

4. Собственно процедура анализа. Процедура должна завершиться получением годных для интерпретации результатов, например, числовых значений параметров.

5. Формулирование выводов. Во время этого этапа нужно на основе полученных результатов сформулировать выводы, соответствующие целям анализа.

Один из самых простых и действенных методов анализа текстов — частотный анализ. Этот метод подразумевает подсчет количества (или относительной частоты) упоминаний заданных понятий в тексте и выводы на основе полученных данных.

С помощью этого метода можно сделать выводы о направленности текста, сделанных в нем акцентах и т.п. Разумеется, не все тексты пригодны для такого анализа. Рассмотрим пример:

1. Цель — проанализировать серию художественных произведений для выявления главных и второстепенных персонажей.

2. Объектом анализа будет набор текстов в виде текстовых файлов, соответствующих правилам машинописной записи и приведенным выше определениям слов, предложений, знаков препинания. Анализ будет проводиться на основе имен собственных — имен людей.

3. Метод анализа — частотный анализ имен собственных. С достаточно высокой степенью уверенности можно сказать, что чем чаще герой будет упоминаться в тексте, тем большую роль в повествовании он играет. В простейшем случае мы можем ограничиться подсчетом прямых упоминаний (хотя можно сформулировать ряд правил для выделения и косвенных упоминаний).

4. Подготовка материалов — обработка текстов по вышеописанным правилам.

5. Процедура анализа будет выглядеть так: мы выделим имена собственные, уберем все, что не относится к именам людей, учтем склонения имен (приведем все к единой форме) и сгруппируем их с помощью любого средства.

6. Выводы мы можем сделать на основе как общего количества упоминаний, так и упоминания героев в смысловых отрывках (например, в главах). Главные герои будут упоминаться по всему тексту, эпизодические — в некоторых отрывках.

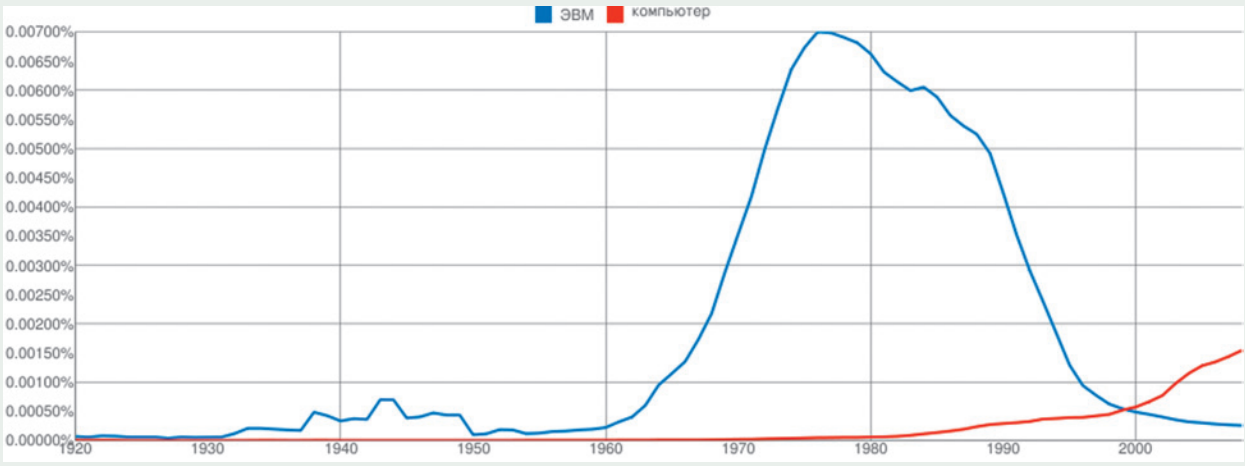


Рис. 1. Пример анализа текста

На *рис.* 1 приведены графики относительной упоминаемости[3] слов “ЭВМ” и “компьютер” в русскоязычных книгах за период с 1920-го по 2010 год. Из графиков можно уверенно предположить, что сокращение “ЭВМ” упоминается все реже и постепенно меняется на слово “компьютер”.

Конечно, анализ художественного текста таким способом вряд ли будет представлять серьезную практическую ценность. Но вот проведенный таким способом анализ группы новостных сообщений уже может быть практически полезен, например, для того чтобы определить, какие факторы (или субъекты событий) влияют на их формулировки.

Частотный анализ, помимо прочего, позволяет нам судить — относится ли текст к какой-то заранее заданной группе текстов[4]. При этом опираются на эмпирически выведенные закономерности, получившие название *закономерности Зипфа*[5].

Первый закон Зипфа:

Произведение вероятности обнаружения слова в тексте на его ранг частоты (порядковый номер при сортировке по убыванию частоты) — константа[6].

Второй закон Зипфа:

Форма кривой зависимости частоты и количества слов, входящих в текст с этой частотой, одинакова для всех текстов.

То есть практически на любом достаточно большом объеме текста слово с рангом **2** встречается приблизительно в два раза реже, чем слово с рангом **1**. Слово с рангом **3** — в три раза реже слова с рангом **1** и т.д. При обработке группы текстов чаще всего подсчитывают значение частоты на миллион словоформ.

Эти закономерности, подмеченные изначально без всякой автоматической обработки, позволили обосновать следующие правила:

1. Наиболее часто встречающиеся в текстах слова не несут характеристической, смысловой нагрузки.

2. Для достаточно устойчивых типов текстов можно вывести частотные характеристики (фактически — относительный порядок) специфических слов. Эти слова будут находиться примерно “в середине” частотной кривой (часто встречающиеся слова — предлоги и т.п., редко встречающиеся — уникальные слова).

Эти эвристические правила активно используются в поисковых системах, системах классификации и других системах автоматической обработки текстов.

На основе частотного анализа построено несколько мощных методов анализа текстов на естественном языке.

Контент-анализ. Суть этого метода состоит в оценке общего направления действий (или вычленения текстов, соответствующих данному направлению) на основе придания объектам (людям, местам, событиям) определенной “окраски” в тексте. “Окраску”, как правило, определяют на основе прилагательных, устойчиво связываемых с этими элементами (точнее — со словами, их обозначающими). Ручной контент-анализ — трудоемкая задача, чреватая большим количеством ошибок. Автоматические средства выполнения такого анализа должны не только вычленять отдельные слова, но и определять структуру предложений.

Существенным этапом при выполнении такого анализа является формирование специализированного словаря-тезауруса, с помощью которого можно оценить “окраску” того или иного упоминания. Такая оценка зависит не только от простого упоминания слова, но и от целевой аудитории текста, времени его написания, способа употребления. Составление такого словаря — достаточно субъективная задача, но законы Зипфа упрощают ее решение.

---

[3] Данные сервиса *http://ngrams.googlelabs.com/* — результат анализа книг в хранилище Google Books.

[4] Характеристический набор текстов для языка или вида текстов называется *корпусом текстов*.

[5] Выведены и предложены профессором-лингвистом Гарвардского университета Джорджем Кингсли Зипфом (George Kingsley Zipf) в 1949 году.

[6] Для русского языка — примерно 0,06–0,07, для английского — примерно 0,1.

Контент-анализ применяется и в более упрощенном виде, например, для определения общего содержания сообщения, для оценки изменчивости в группе сообщений и т.д. Особую ценность этот метод имеет в связи с тем, что это один из немногих способов применить точные, количественные методы анализа и доказательства в традиционно гуманитарных областях знания.

Практикум

В практической части мы решим несколько задач, которые основаны на автоматической обработке текста. Поскольку задачи обработки текста очень разнообразны, мы не можем, конечно, утверждать, что показанный нами инструментарий — самый лучший и единственно верный. Тем не менее у нас есть основания утверждать, что он достаточно мощен, чтобы решать с его помощью массу полезных задач.

Основной инструментарий

Основным средством, которое мы будем использовать для обработки текста, будет реализация языка Pascal — PascalABC.Net. Напомним, что получить его можно по адресу:

http://pascalabc.net/ssyilki-dlya-skachivaniya.html.

Помимо рабочей среды языка, нам потребуется программа подготовки и тестирования регулярных выражений Expresso. Программа эта бесплатная, и ее можно скачать по адресу:

http://www.ultrapico.com/ExpressoDownload.htm.

В этом разделе мы будем опираться на возможности уже не столько самой системы, сколько на возможности дополнительных библиотек — как стандартных для платформы .Net, так и дополнительно полученных.

Поскольку авторы ограничены “газетным” вариантом — некоторые вопросы (такие, как, например, использование лингвистических средств) в этот вариант практикума не войдут.

Выделение последовательностей по шаблону

Задача — найти в тексте часть, соответствующую шаблону (или проверить, соответствует ли строка шаблону), возникает очень часто и в самых разных ситуациях. Основным способом ее решения, как мы уже говорили в учебнике, является механизм регулярных выражений.

Основным средством, которое мы будем использовать для создания и проверки регулярных выражений, будет программа Expresso. Начнем с краткого разбора ее интерфейса (*рис.* 2).

Основное рабочее окно программы — набор из трех страниц: Test Mode (режим проверки), Design Mode (режим создания) и Expression Library (библиотека выражений). В основном мы будем работать на первой — в режиме тестирования.

На этой странице есть, во-первых, управляющая панель, а во-вторых, несколько разделов:

- Regular Expression — само регулярное выражение. В этом месте мы можем написать выражение — просто как текст;
- Regex Analyzer — анализатор выражения. В этом окне выражение будет разобрано на части так, как его будет интерпретировать механизм. Там можно проверить синтаксис. Если выражение сложное, то будет показано дерево — чтобы можно было проверить вложение частей;
- Sample Text — пример текста. В этот раздел загружается текст, к которому и применяется регулярное выражение. Текст можно просто скопировать туда;
- Search Result — раздел, в котором будет показан результат применения — т.е. поиска — регулярного выражения. Там будут показаны все найденные вхождения, и если вхождение выбрать в примере текста, будет показано, где именно оно найдено. Если выражение включает в себя несколько групп, то результат можно развернуть — опять-таки, как дерево.

Решим несколько задач, опираясь на сведения о синтаксисе регулярных выражений. Во всех задачах нам нужно будет написать выражение, проверить его, выполнить и оценить результат.

Задача 1. Выбрать из текста все целые числа.

Начнем с того, что выясним — из чего может состоять целое число? Очевидно, в основном — из

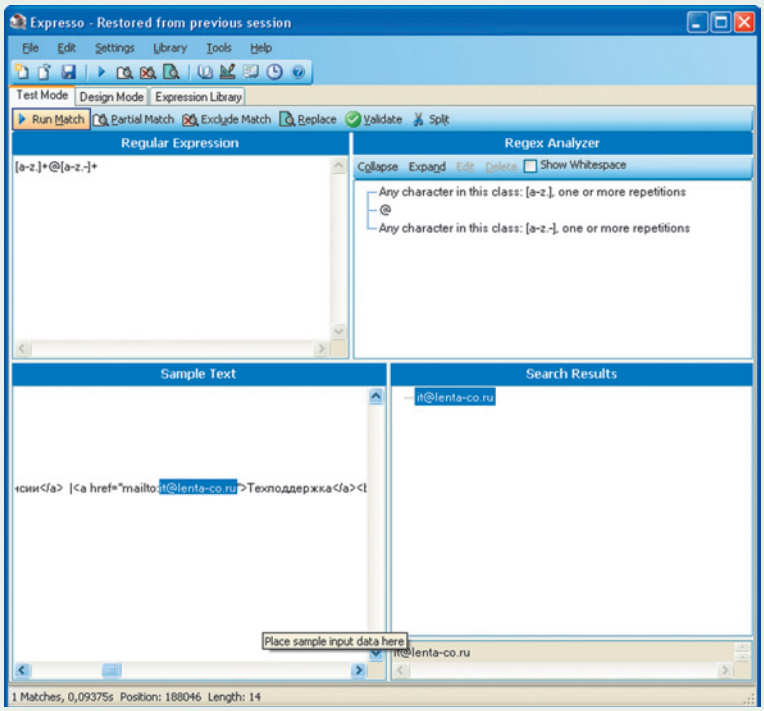


Рис. 2

цифр. Кроме этого, прямо перед числом может быть знак “–” или “+” (а может их и не быть). Кроме этого, последовательность цифр может стоять в начале или в конце отдельного слова, — но такая конструкция (например, “Иванов-2й”) числом не будет.

Используем для этого выражения символьные классы и символы повторения:

```
[+-]?\d+
```

Выражение очень простое: первый его символ — это либо “+”, либо “–”, это мы и записали в его классе. Он может возникнуть один раз, а может не возникнуть — поэтому выбран квантификатор “?”, то есть необязательный элемент.

Следом идет набор цифр (символьный класс `\d`), причем в нем минимум одна цифра.

Применив это выражение, мы выясним, что оно действительно находит все правильно написанные числа, но ситуацию “Иванов-2й” обрабатывает неверно — выделяет из этой подстроки число 2.

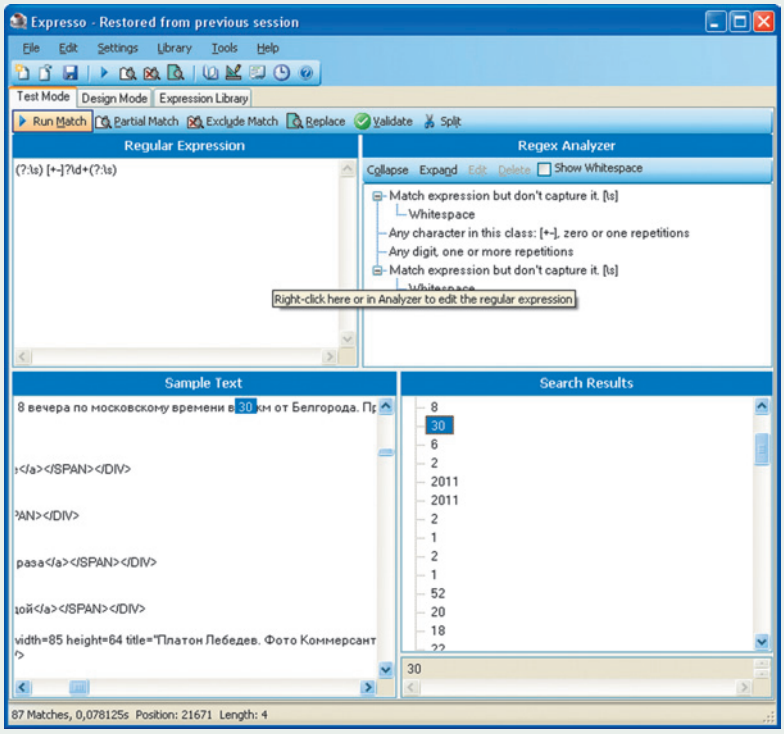


Рис. 3

Чтобы исключить эти ситуации, мы воспользуемся так называемой *незахватывающей проверкой*: то есть конструкцией, которая проверит часть строки на соответствие шаблону, но не выделит эту часть строки из результата. Синтаксис этой конструкции таков: `(?:<выражение>)` — мы проверяем, что `<выражение>` есть, но не включаем его в результат. Получится примерно вот что:

```
(?:\s)[+-]?\d+(?:\s)
```

То есть мы потребовали, чтобы перед и после нашего числа стояли пробельные элементы (класс `\s`).

Вот что мы получили при тестировании, просто записав HTML-код страницы в виде текста (см. *рис.* 3).

Обратите внимание: в разделе Regex Analyzer подробно разобрано, в каком порядке и как анализируется текст в этом регулярном выражении.

Внимательно посмотрев, что именно выделено, мы поймем, что нашли не только нужное, но и пробелы вокруг. Чтобы все-таки выделить только целое число, придется заключить нужную часть в круглые скобки — тогда первая группа внутри найденного совпадения как раз и будет нужным нам числом:

```
(?:\s)([+-]?\d+)(?:\s)
```

Теперь рядом с каждым совпадением появится знак развертывания ветки — в которой и будет нужная группа.

Задания для самостоятельной работы

1. Напишите выражение, которое найдет числа в составе слов.

2. Напишите выражение для поиска чисел в формате числа с плавающей точкой.

3. Напишите выражение для поиска телефонных номеров.

Задача 2. Выбрать из текста все упоминания весов в метрической системе (например, 5 килограмм, 10 тонн и т.д.). Для упрощения задачи считать, что все веса — целые, а меры написаны без сокращений.

Определим структуру текста, который нам нужно найти: он состоит из числа и меры, которая за ним записана. Как искать число — мы разобрали в прошлой задаче.

Кратное	Название	Кратное (СИ)	Название	Кратное	Название	Кратное (СИ)	Название
10^0	**тонна**	10^6	мегаграмм	10^0	**тонна**	10^6	мегаграмм
10^1		10^7	*(нет)*	10^{-1}	децитонна		
(10^2)	(центнер)	10^5	*(нет)*				
10^2		10^8	*(нет)*	10^{-2}	сентитонна	10^4	*(нет)*
10^3	килотонна	10^9	гигаграмм	10^{-3}	миллитонна	10^3	килограмм
10^6	мегатонна	10^{12}	тераграмм	10^{-6}	микротонна	10^0	грамм
						10^{-3}	миллиграмм

В метрической системе мер меры образуются из основного слова и десятичной приставки. Все десятичные приставки мы использовать не будем, но некоторые перечислим в таблице.

Чтобы реализовать эту структуру, мы используем выбор из нескольких вариантов:

```
(?:\s)\d+)(?:\s)+(((?:кило|мега|
гига|тера|санти|милли|сенти|санти|
микро)?(?:грамм|тонн))|(г|см|м|км))
```

Напомним, что знак вопроса после группы задает возможное (но не обязательное) появление.

Как видите, кроме десятичных приставок мы добавили сокращения — на них правила записи не распространяются, поэтому придется перечислить их.

Применив это выражение, мы выясняем, что оно действительно выбирает необходимое, но не только (*рис.* 4).

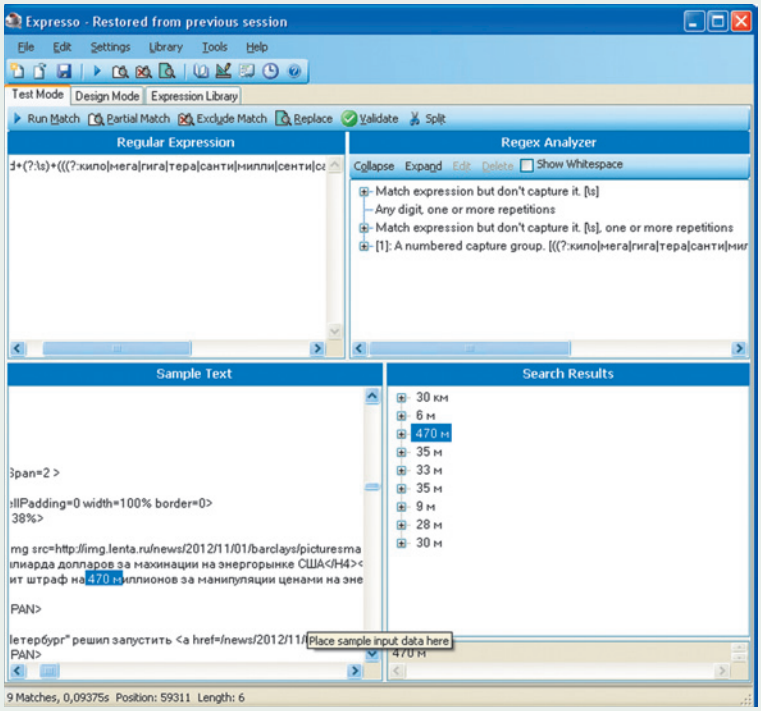


Рис. 4

470 м — записано как мера веса, но речь в тексте — о размере штрафа. Поэтому нам придется добавить дополнительное условие в конец: расширенное множество символов:

```
(?:\s)\d+(?:\s)+(((?:кило|мега|
гига|тера|санти|милли|сенти|санти|
микро)?(?:грамм|тонн))|(г|см|м|км))
[ .,:;!?{}()\\[\]]+
```

Задания для самостоятельной работы

1. Внесите изменения так, чтобы из найденного можно было выделить ТОЛЬКО количество и меру веса.

2. Напишите аналогичное выражение для выбора мер расстояния.

Задача 3. Выбрать из текста все адреса электронной почты.

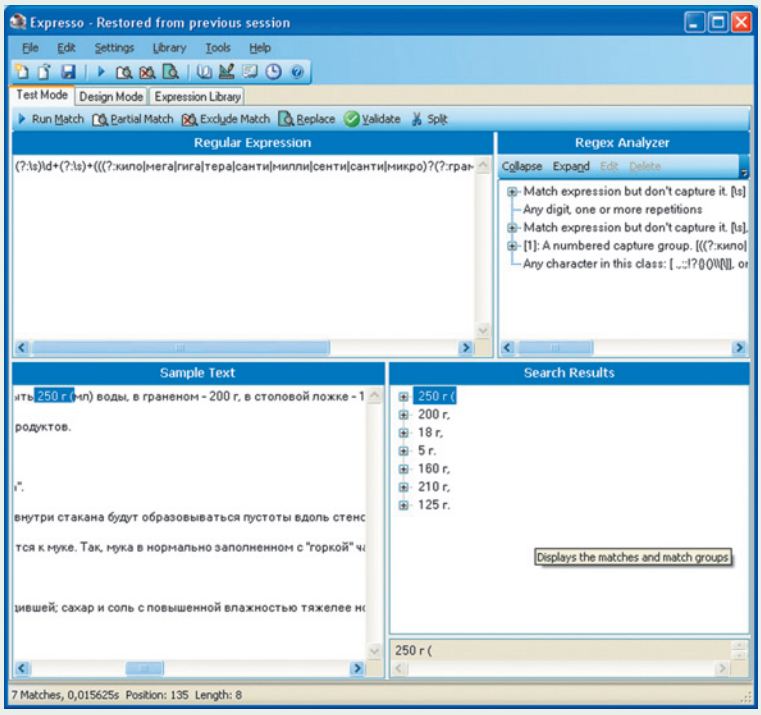


Рис. 5

Как обычно, первое, что мы делаем, — это выясняем: из каких символов состоит адрес? Согласно стандарту, адрес может состоять из двух частей: имени пользователя и имени почтового отделения. Имена разделяются в адресе знаком “@”. Так что этот знак будет обязательно. Имена и слева и справа могут состоять из символов латинского алфавита, цифр, точки, знака тире и подчеркивания. Регистр букв имеет значение.

Стандарт допускает использование и ряда других символов, но на нашу работу это повлияет мало.

Поэтому наше выражение будет состоять из трех частей:

1. Множества символов — названия ящика. Причем название не пустое — минимум один символ из этого множества должен быть.

2. Знака “@”.

3. Множества символов — названия почтового отделения.

Множество символов запишем так:

```
[A-Za-z0-9.-_].
```

Поскольку оно встречается минимум один раз, то зададим повтор знаком “+”.

В итоге выйдет примерно следующее:

```
[A-Za-z0-9.-_]+@[ A-Za-z0-9.-_]+
```

Для проверки скопируем реальный текст в окно примера и применим выражение. Текст мы взяли из html-кода страницы сервера ново-

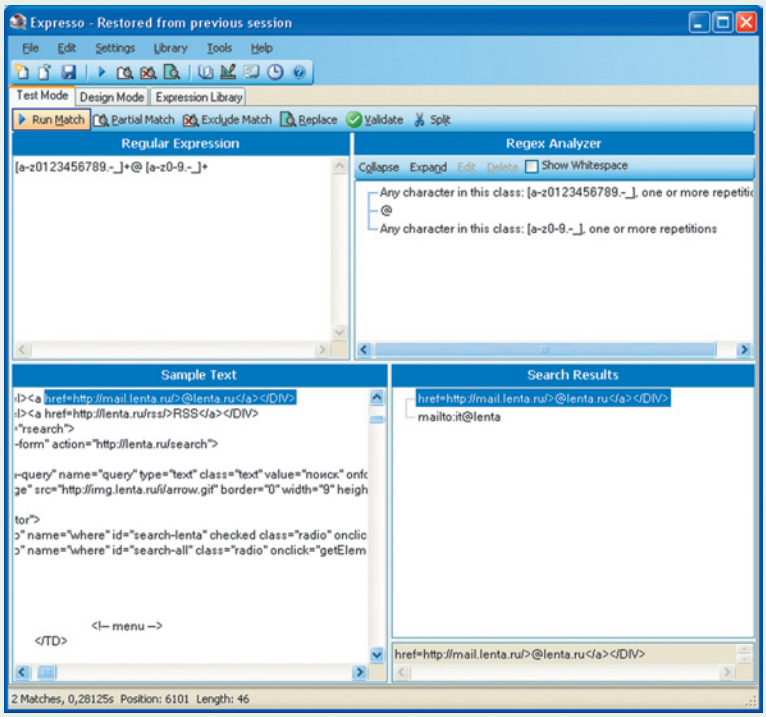


Рис. 6

стей и, применив, увидели примерно вот что — см. *рис.* 6.

Как видим, адреса выделены неправильно. Причина — появились лишние символы-разделители. Откуда они взялись? Проанализируем запись множества и обнаружим, что добавили лишний диапазон: .-_. Мы имели в виду просто символ "-", но он имеет в записи множеств специальный смысл. Поэтому перед его упоминанием ставим знак "\".

Теперь результат будет верен, он станет примерно таким, как на первой иллюстрации. Выражение будет выглядеть так:

```
[A-Za-z0123456789.\-_]+@[ A-Za-z0-9.\-_]+
```

Тем не менее этого недостаточно. Название почтового ящика не может начинаться с точки и содержать две точки подряд. После знака-разделителя тоже не может стоять точка.

Решим эту задачу так: разобьем последовательность на группы:

```
[A-Za-z0123456789\-_]+(?:\.
[A-Za-z0123456789\-_])*@[A-
Za-z0123456789_]+(?:\.[A-Za-
z0123456789\-_])+
```

Обратите внимание: в обоих случаях в начале идет группа без точки, а потом некоторое количество групп, которые начинаются с точки. Если в первой части таких групп может и не быть, то во второй части они есть обязательно.

Задания для самостоятельной работы

1. Напишите регулярное выражение для выборки из текста дат в формате ДД/ММ/ГГ или ДД/ММ/ГГГГ. В выражении учтите, что в датах не бывает номера дня больше 31 и месяцев всего 12.

2. Вы получили текстовый файл протокола обращений пользователей к ресурсам среды Интернет, в котором построчно перечислены атрибуты каждого обращения через запятую. Порядок полей совпадает, названия полей заданы в первой строке. Подготовьте регулярное выражение, позволяющее отфильтровать только 1-, 4-й и 7-й столбцы и перечисляющее значения через точку с запятой.

3. Выделите с помощью регулярного выражения все имена собственные из географического текста (например, реферата).

Использование регулярных выражений при подготовке программ

Теперь рассмотрим, каким образом мы сможем использовать механизм регулярных выражений в своих программах. Стандарт Паскаля не предусматривает библиотек для работы с ними, но в учебной среде нам доступна библиотека, входящая в комплект платформы .Net. Подключим ее в модуле Uses и получим возможность использовать объект Regex и необходимый тип Match — совпадение.

```
program RegularExperssionsMatch;
uses System.Text.RegularExpressions;
var
  s: String;
  txt: Text;
  m: Match;
begin
  Assign(txt, 'k:\demo.txt');
  Reset(txt);
  while not Eof(txt) do
  begin
    ReadLn(txt, s);
    foreach m in Regex.Matches(s,
        '[A-Za-z0123456789\-_]+
         (?:\.[A-Za-z0123456789\-_])*
         @[A-Za-z0123456789_]+
         (?:\.[A-Za-z0123456789\-_])+') do
    Writeln('Найдено:', m.Value);
  end;
  Close(txt);
end.
```

Обратите внимание на конструкцию **foreach**... **in**... — с ее помощью мы получаем возможность перебрать все элементы во множестве-коллекции. Эта синтаксическая конструкция отсутству-

ет в классическом языке Паскаль, но есть в расширенном (Extended Pascal, ISO 10206:1990).

Задания для самостоятельной работы

1. Напишите программу поиска заданного слова в текстовом файле. Считайте, что текст записывается без использования переносов.

2. Напишите программу для поиска двух идущих подряд заданных слов. Учтите, что между словами может быть несколько разделителей — например, два и более пробелов.

3. Напишите программу, подсчитывающую количество упоминаний в текстовом файле семьи Ивановых (или любого из ее членов). Считайте, что фамилия во всей семье одна.

Помимо поиска совпадений, регулярное выражение можно использовать для разделения строки на части. В этом случае регулярное выражение будет использоваться для поиска последовательностей-разделителей.

В этом примере с помощью этой возможности строки делятся на слова — то есть последовательности символов между последовательностями разделителей.

```
Program RegularExpressionsSplit;
uses System.Text.RegularExpressions;
var
 s,s0: String;
 txt: Text;
begin
  Assign(txt, 'k:\demo.txt');
  Reset(txt);
  while not Eof(txt) do
  begin
    ReadLn(txt, s);
    foreach s0 in Regex.Split(s, '[ ,.!?{}
()\[\]/*+\-]+') do
      Writeln(s0);
  end;
  Close(txt);
end.
```

Задания для самостоятельной работы

1. Напишите программу подсчета общего количества слов в текстовом файле. Считайте, что текст записывается без использования переносов, числа не считаются словами.

2. Напишите программу обработки текстового файла с переносами, которая уберет переносы — то есть восстановит разбитые слова и результат сохранит в файл, добавив к имени `.new`. Учтите, что знак, обозначающий перенос, может применяться и в других случаях.

Еще один метод, который нам может понадобиться, — замены. Этому методу нужно будет передать второе выражение, которое определит, что должно оказаться на месте найденного фрагмента.

Приведем пример, в котором по всему текстовому файлу заменим квадратные скобки на угловые.

```
Program RegularExperssionsReplace;
uses System.Text.RegularExpressions;
var
 s,s0: String;
 txt: Text;
begin
  Assign(txt, 'k:\demo.txt');
  Reset(txt);

  while not Eof(txt) do
  begin
    ReadLn(txt, s);
   s0 := Regex.Replace(s, '\[.+\]', '<$0>');
    Writeln(s0);
  end;
  Close(txt);
end.
```

Задания для самостоятельной работы

1. Вы получили текст, в котором все даты записаны в виде ГГГГ-ММ-ДД (например: 2010-12-31 — 31 декабря 2010 года), или ДД-ММ-ГГ, или ДД-ММ-ГГГГ. Напишите программу, которая преобразует все эти даты к привычному нам виду: ДД.ММ.ГГГГ (т.е. 31.12.2010) — и результат сохранит в новый текстовый файл.

2. Вы получили текст, в котором записаны номера телефонов в разных формах. Примеры из текста: +7 (495) 123-4567, 123-4567, 1234567, 499-1234567. Напишите программу, которая преобразует все эти варианты к общему стандартному: +7 (495) 123-4567 — и результат сохранит в новый текстовый файл. По умолчанию (если не указано) предполагаем, что номера российские, код города соответствует коду вашего города.

Частотный анализ

Идея частотного анализа очень проста: если у нас есть набор каких-то элементов (букв, слов, предложений, яблок — чего угодно), то простой способ оценить этот набор — посчитать, сколько в этом наборе элементов каждого вида (каждой буквы, слова, размера яблок). Узнав, какую долю (то есть насколько часто встречается) занимает элемент, мы можем делать какие-то выводы.

Анализ частоты символов в тексте

Первая группа задач, которую мы будем решать, будет связана с кодировками и алфавитом.

Дано: текст на русском языке, в кодировке Windows-1251, длина заранее не известна, в форме файла.

Требуется: составить частотную таблицу упоминаемости символов в этом тексте.

Решение

Нетрудно увидеть, что основной частью этой задачи будет посчитать количество упоминаний каждого символа в тексте. Вспомним определение таблицы кодировки и особенности конкретной кодировки Windows-1251. Нам точно известно, что в ней всего 256 символов, с номерами от 0 до 255 включительно.

Создадим массив из 256 беззнаковых (не может быть отрицательного количества) целых чисел. Будем считывать по одному символу и увеличивать значение в соответствующей (то есть с номером, равным коду символа) ячейке массива. Единственное, что нам придется сделать, — преобразовать символ в его код.

Заготовка программы будет выглядеть примерно так:

```
program frequency;
var
  txt: Text;
  c: char;
  countLetters: array[0..255] of Word;
  i: Word;
begin
  Assign(txt,'demo.txt');
  Reset(txt);
  for i := 0 to 255 do
    countLetters[i] := 0;
  while not Eof(txt) do
  begin
    read(txt,c);
    inc(countLetters[ord(c)]);
  end;
  for i := 0 to 255 do
    WriteLn( i, ' — ',countLetters[i]);
  Close(txt);
end.
```

Найдем какой-нибудь достаточно большой текстовый файл, сохраним его под именем *demo.txt* и посмотрим, что получится.

Проанализировав результаты, мы выясним, что:

1. Довольно много символов не относится к языку, а часть — даже не буквы (особенно символы “возврат каретки” и “перевод строки”, “пробел” и т.п.). Очевидно, они не должны влиять на частоту.

2. Если объем текста достаточно велик (около 3 Мб), появляется довольно большая вероятность получить ошибку “Переполнение”.

3. Поскольку нам придется считать общее количество символов, то переменная для этого точно должна допускать большие значения, чем обычный тип Word.

При подготовке программы нам нужно будет ввести фильтрацию символов. Самый простой способ, если мы работаем с русским языком, — учитывать только символы расширенного набора — т.е. с кодами больше 255.

В итоге программа будет примерно такой:

```
program frequency;
var
  txt: Text;
  c: char;
  countLetters: array[0..255] of LongWord;
  i: Word;
  sigma: LongWord;
begin
  Assign(txt,'demo.txt');
  Reset(txt);
  for i := 0 to 255 do
    countLetters[i] := 0;
  while not Eof(txt) do
  begin
    read(txt,c);
    if ord(c) > 127 then
      inc(countLetters[ord(c)]);
  end;
  sigma := 0;
  for i := 0 to 255 do
    sigma := sigma + countLetters[i];
  for i := 128 to 255 do
    WriteLn(chr(i), '(',i, ')', ' — ',
            (countLetters[i]/sigma):8:7);
end.
```

Результат запуска программы для текста 1-го тома романа “Война и мир” приведен в таблице на с. 29.

Несколько вопросов и предложений для совершенствования программы:

1. Проанализируйте вывод программы. Можно ли сделать результат ее работы более коротким, если ограничиться русским языком?

2. Можно ли сократить объем занимаемой памяти?

3. Стоит ли вместо отдельного цикла с подсчетом количества русских букв добавлять единицу к сумме сразу в основном цикле?

4. Что еще нужно учесть, чтобы получить настоящую частотную таблицу — т.е. таблицу, в которой частота подсчитана именно *для буквы*, а не *печатного символа*?

5. Предложите вариант усовершенствования, который сможет работать с кодировкой Unicode.

Очевидно, что частоты встречаемости символов в больших русских текстах хоть немного, но будут различаться. Но вот порядок символов по частоте упоминаемости будет примерно одинаковым.

Задания для самостоятельной работы

1. Подсчитайте частоты для большого объема текстов разного вида: для официальных документов, литературных произведений. Совпадают ли они? Сохраняется ли порядок символов по частоте встречаемости?

2. Как с помощью частот встречаемости различать таблицу кодировок текста: CP866, Windows-1251, KOI-8?

Буква	Частота	Буква	Частота
о(238)	0.1123347	В(194)	0.0020581
а(224)	0.0822470	А(192)	0.0019786
е(229)	0.0789770	П(207)	0.0019214
и(232)	0.0653834	К(202)	0.0017662
н(237)	0.0635027	О(206)	0.0015796
т(242)	0.0557268	Б(193)	0.0015168
с(241)	0.0524678	М(204)	0.0011122
л(235)	0.0501234	Д(196)	0.0010235
р(240)	0.0445219	С(209)	0.0008868
в(226)	0.0437774	Р(208)	0.0008591
к(234)	0.0346491	Т(210)	0.0008406
д(228)	0.0292490	И(200)	0.0008018
м(236)	0.0283326	Я(223)	0.0005616
у(243)	0.0283308	ъ(250)	0.0005247
п(239)	0.0236641	Г(195)	0.0004896
я(255)	0.0224928	Э(221)	0.0004360
г(227)	0.0201539	Ч(215)	0.0003991
ь(252)	0.0193780	Е(197)	0.0003399
ы(251)	0.0188958	Л(203)	0.0002697
з(231)	0.0175231	З(199)	0.0002051
б(225)	0.0156794	У(211)	0.0002014
ч(247)	0.0131742	Ж(198)	0.0001496
й(233)	0.0114783	Ш(216)	0.0001275
ж(230)	0.0099319	Ф(212)	0.0001256
ш(248)	0.0092742	Х(213)	0.0001127
х(245)	0.0083801	Ц(214)	0.0000388
ю(254)	0.0064421	Ю(222)	0.0000148
ц(246)	0.0039979	Ь(220)	0.0000074
щ(249)	0.0027897	Щ(217)	0.0000037
э(253)	0.0025735	Й(201)	0.0000018
ф(244)	0.0021024	Ъ(218)	0.0000000
Н(205)	0.0020913	Ы(219)	0.0000000

3. Напишите программу преобразования текста между разными таблицами кодировки:

a) Win1251 <-> KOI-8

b) Win1251 <-> CP866

c) CP866 <-> KOI-8

Анализ частоты упоминаемости слов в тексте

Дано: текст на русском языке, в кодировке Windows-1251, длина заранее не известна, в форме файла.

Требуется: составить частотную таблицу упоминаемости слов в этом тексте.

Решение

При решении этой задачи нам нужно учесть несколько особенностей:

- во-первых, мы не знаем, сколько слов в тексте будет;
- во-вторых, в отличие от символов слово нужно в тексте выделить, потому что это не единый объект, а набор объектов;
- в-третьих, слово может быть написано поразному, в зависимости от того — первое это слово в предложении или нет;
- слово может быть разделено на части: если текст заранее отформатирован устаревшими методами.

На самом деле трудностей больше, но решать их мы будем позже.

Выделять слова из текста мы будем с помощью регулярного выражения. Чтобы его построить, мы вспомним, что такое “слово”. Нас будет интересовать строго формальное определение, которое позволит нам выделить слово из потока символов. С этой точки зрения слово — это последовательность символов, взятых из текста, и не являющихся разделителями.

Воспользуемся для создания программы, во-первых, методом деления строки на части по регулярному выражению, а во-вторых, библиотекой работы с коллекциями из .Net для организации словаря:

```
Program Frequency;
uses System.Text.RegularExpressions,
     System.Collections.Generic;
var
 s,s0: String;
 txt: Text;
 tx: System.Collections.Generic.
     Dictionary<string, integer>;
 m: Match;
begin
  tx:= new System.Collections.Generic.
           Dictionary<string, integer>;
  Assign(txt,'demo.txt');
  Reset(txt);
  while not Eof(txt) do
  begin
    ReadLn(txt,s);
    foreach s0 in Regex.Split(s, '[,.!?()
    {}+*\-;:"\[\]\\]+') do
      if s0 <> '' then
      if tx.ContainsKey(s0) then
          tx[s0] := tx[s0]+1
        else
          tx.Add(s0,1);
  end;
  close(txt);
  Assign(txt,'k:\frequency.csv');
  Rewrite(txt);

  WriteLn( txt,'Word;Count;' );
  foreach s0 in tx.Keys do
    WriteLn( txt,s0, ';', tx[s0],';' );
  Close(txt);
end.
```

Запустив эту программу, мы сможем проанализировать текстовый файл и получить результаты примерно в таком виде, как приведено в таблице на с. 30.

Изначально результат будет не отсортирован, но это сделать несложно.

Автор	2
автор	8
автора	3
авторе	1
авторитет	9
авторитета	2
авторитетами	1
авторитете	1
авторитетная	1
авторитетнее	1
авторитетной	1
авторитетную	1
авторитетом	1
авторитету	3
авторитеты	1
авторов	2
авторских	1
авторскую	1
авторство	1
автору	3
авторы	3
Авторы	1

Ага	16
агитации	2
агрегата	1
агрегаты	1
агронома	1
Административная	1
административная	1
административной	1
административную	2
административные	1
административный	1
администратор	2
администратора	1
администрацией	2
администрации	1
администрировании	1
администрировать	1

На что нам придется обратить внимание, прежде чем мы сможем воспользоваться результатами для анализа? На несколько особенностей:

1. Учитывается регистр, а значит, “Административная” и “административная” — это два разных слова. Очевидно, что с точки зрения извлечения смысла это не так.

2. Самые частые слова не несут никакого значения для анализа текста. Сложно предположить, что нам важна частота слова “Ага” или “И”.

3. Мы не учитываем формы слов, поэтому, например, видим 6 вариантов одного и того же по смыслу слова.

Первая проблема решается довольно просто: нужно перед разбиением преобразовать всю строку к единому регистру — верхнему или нижнему. Метод, который это реализует, — метод самой строки `toLower` (или `toUpper` — логически разницы никакой нет). Добавим в программу:

```
s := s.ToLower;
```

Куда это поместить в программе — надеемся, читатель в состоянии определить сам.

Вторая проблема решается несколько труднее: нельзя просто выкинуть слово, потому что оно часто встречается в этом конкретном тексте. Чтобы понять, какие слова не характеризуют конкретный текст, нам нужно проанализировать большое количество эталонных текстов — выделить самые частые слова. К счастью, за нас это уже сделано: воспользуемся статистической информацией с сайта корпуса русского языка: раздел “Частоты”, самые частые 100 словоформ будут нашим стоплистом: *http://ruscorpora.ru/1grams.top.html*.

Для реализации нужно будет сохранить их в файл, а в программе предусмотреть отдельную коллекцию стоп-слов, в которую загрузить слова из файла и проверять перед добавлением — есть там слово или нет.

Реализацию этого подхода авторы предоставляют читателям для самостоятельного упражнения.

Третья проблема существенно сложнее: для ее решения нам придется как минимум преобразовать слова к их исходной, базовой форме, т.е. выполнить так называемую *лемматизацию*. Для русского языка это непростая задача, которую можно решить использованием специализированной библиотеки. Как было сказано вначале, в рамках сокращенного газетного варианта делать это было бы очень затруднительно — и для авторов, и для читателей.

Автоматизированная и автоматическая обработка текста, извлечение из текста данных, анализ больших массивов информации, поиск и смежные с ним задачи — современное и быстро развивающееся направление. Его достижениями мы пользуемся постоянно — когда обращаемся к любым документальным и поисковым системам, работаем со сложными сайтами. Авторы полагают, что ознакомление с базовыми приемами и средствами такой работы — необходимый элемент подготовки в области информационных технологий.

Красота этой картины завораживает. Есть в ней что-то загадочное, непостижимое и даже космическое. Это один из аттракторов, построенный на компьютере. Аттракторы — это множества, возникающие в теории динамических систем. Разрабатывая сложные математические структуры, ученые создают уникальные модели, пытаясь ответить на многочисленные вопросы, поставленные природой

Эффект порхающей бабочки

А.И. Азевич,
к. п. н.,
доцент кафедры информатизации образования МГПУ,
Москва

Чтобы понять, как возникли эти таинственные рисунки, начнем с небольшого экскурса в историю. Наверняка многие слышали о дифференциальных уравнениях. Впервые они встречаются в работах великих творцов новой математики И.Ньютона и Г.Лейбница. Изучение этих вопросов было неслучайным, ведь благодаря дифференциальным уравнениям можно описать многие природные явления. А к этому люди стремились испокон веков.

Очень часто физические законы устанавливают соотношения между некоторыми величинами и скоростями их изменения. А они, в свою очередь, входят в равенства, содержащие как функции, так и их производные. Именно в ходе изучения реальных процессов возникает необходимость в дифференциальных уравнениях.

Рассмотрим конкретную задачу, в которой надо определить закон изменения некоторой величины. Пусть какая-то металлическая пластина, нагретая до температуры y_0, в момент времени $t = 0$ погружается в большой сосуд, наполненный воздухом нулевой температуры. С течением времени пластина начинает охлаждаться, а ее температура становится функцией времени $y(t)$. По закону охлаждения И.Ньютона, скорость изменения температуры пластины, т.е. производная $\frac{dy}{dt}$, пропорциональна разности температур тела и окружающей среды. А так как по условию задачи температура окружающей среды равна нулю, то

$$\frac{dy}{dt} = -ky. \qquad (1)$$

Здесь k — положительный коэффициент, характеризующий свойства материала, из которого изготовлено тело (пластина). Знак “минус” в формуле показывает, что температура убывает.

Полученное уравнение — математическая модель процесса охлаждения тела, помещенного в сосуд. Решить уравнение — значит найти все такие функции $y(t)$, которые обращают его в тождество. Для этого проинтегрируем обе части равенства (1).

Получим общее решение дифференциального уравнения:

$$y = Ce^{-kt}. \qquad (2)$$

В процессе охлаждения пластины нас интересует только то решение, которое соответствует моменту времени $t = 0$. Подставив в формулу (2) $t = 0$ и $y = y_0$, получим $C = y_0$. Тогда частное решение дифференциального уравнения примет вид: $y(t) = y_0e^{-kt}$. Получается, что изменение температуры пластины происходит по потенциальному закону. На *рис.* 1 изображен график, показывающий, что с течением времени температура пластины понижается и стремится к температуре окружающей среды. В данном случае пластину окружает воздух, температура которого равна 0.

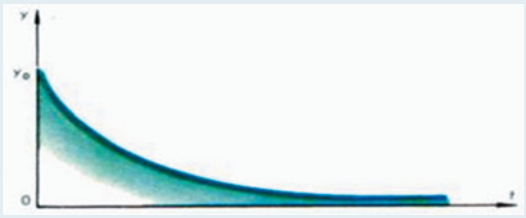


Рис. 1

Мы привели простой пример, связанный с решением дифференциального уравнения. Вернемся к рисунку аттрактора и рассмотрим его подробнее.

Что же такое аттрактор? Аттрактор (от англ. *to attract* — притягивать) — это особое множество, возникающее в теории динамических систем. Но что понимают под динамической системой? Обычно ее представляют как некоторое геометрическое пространство, на котором задано правило поведения точек во времени. Причем время может быть дискретным (прерывным) или непрерывным. В последнем случае динамические системы задаются системой дифференциальных уравнений, а не одним уравнением, как в предыдущем примере. Ясно, что при построении множества — решений системы — получится более сложная и интересная картина.

Именно такую картину впервые нарисовал американский метеоролог и математик Эдвард Лоренц (1917–2008). Пытливый ученый наблюдал из окна своей научной лаборатории утренний туман и низкие облака, пришедшие с Атлантики. Восхищаясь красотой природных явлений, он пытался построить их компьютерные модели.

Вычислительная машина, сконструированная Э.Лоренцом, была по современным понятиям громоздкой и примитивной. Это нагромождение проводов и вакуумных ламп занимало половину небольшого научного кабинета. Машина была раздражающе шумной и часто ломалась. Устройство не обладало ни быстродействием, ни объемом памяти, необходимым для построения модели атмосферы Земли.

И все же в 1960 г. Э.Лоренц создал мини-модель погоды, которая привела в восторг научный мир. Каждую минуту компьютер выдавал стройные ряды чисел. Они сообщали, что господствующее в данный момент западное направление ветра скоро сменится на северное, потом на южное и вновь на северное. Оцифрованные циклоны медленно кружились по воображаемому глобусу на экране компьютера Э.Лоренца. Узнав о любопытном изобретении, преподаватели и старшекурсники стали заключать между собой пари, пытаясь угадать, какой же будет виртуальная погода в каждый следующий момент времени.

Э.Лоренц применил на практике законы И.Ньютона. Творец машинных моделей верил, что классические законы движения ведут его к более сложным математическим структурам. Постигни их — и ты поймешь не только капризы погоды, но и Вселенную! В этом заключалась философия компьютерного моделирования, которой так увлекся математик и метеоролог.

Э.Лоренц написал компьютерную программу для решения системы дифференциальных уравнений

$$\begin{cases} \dfrac{dx}{dt} = s(-x + y) \\ \dfrac{dy}{dt} = rx - y - xz \\ \dfrac{dz}{dt} = -bz + xy. \end{cases}$$

Параметры, входящие в систему, принимали конкретные значения: $s = -10$, $r = 28$, $b = \dfrac{8}{3}$.

Несмотря на кажущуюся простоту, уравнения выражали чрезвычайно сложную динамическую структуру. Множество решений системы в трехмерном пространстве представляет собой аттрактор, названный в честь открывшего его ученого. Аттрактор моделирует воздушные потоки в атмосфере. Непосвященным кажется, что эти движения вообще не подчиняются никаким математическим законам.

Как позже писал Э.Лоренц в одном американском метеорологическом журнале, он вычислял значения решения системы дифференциальных уравнений в течение длительного времени. Однажды его крайне удивил результат решения, полученный в середине интервала счета. Пришлось повторить вычисления с этого момента. Результаты повторного счета, очевидно, совпали бы с результатами первоначального счета, если бы начальные значения для повторного счета в точности были равны полученным ранее значениям для этого момента времени. Лоренц слегка изменил эти значения, уменьшив число верных десятичных знаков. Ошибки, введенные таким образом, были крайне невелики.

Но самое неожиданное было впереди. Вновь просчитанное решение некоторое время хорошо согласовывалось со старым. Однако по мере счета рас-

хождение возрастало, и постепенно стало ясно, что новое решение вовсе не напоминает старое.

Э.Лоренц вновь проводил скрупулезные вычисления (вероятно, не доверяя компьютеру). Делал он это до тех пор, пока не осознал важность эксперимента. То, что наблюдал ученый, теперь называется существенной зависимостью от начальных условий — основной чертой, присущей хаотической динамике. Именно исследование сложных процессов, происходящих в атмосфере, и привело к понятию аттрактора. Аттрактор — это графическая модель динамического хаоса. В скоплении воздушных потоков, которые, как казалось, не поддаются научному описанию, метеоролог-исследователь открыл динамический (изменяющийся) порядок.

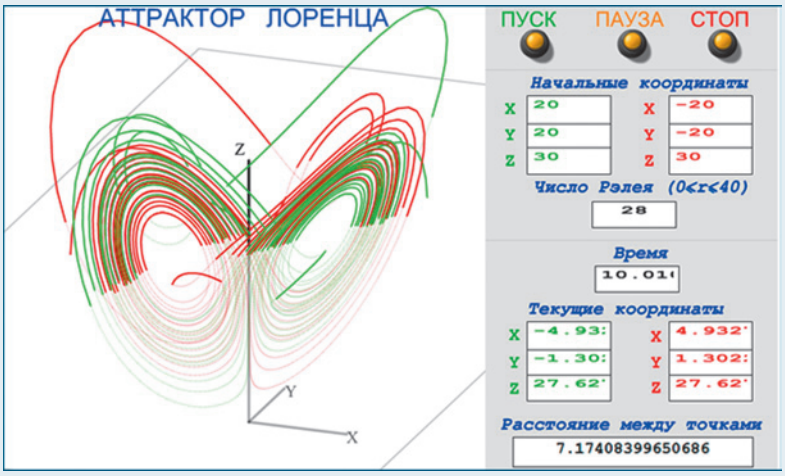


Рис. 3

Зависимость, которую установил Э.Лоренц, называют “эффектом бабочки”. Такое название, возможно, объясняется тем, что, несмотря на совершенные модели и колоссальные расчеты, невозможно сделать долгосрочные прогнозы погоды. Сам Э.Лоренц разъяснил это в статье “Предсказуемость: может ли взмах крылышек бабочки в Бразилии привести к образованию торнадо в Техасе?”, опубликованной в 1979 году.

Его аттрактор — это графическое представление решений системы дифференциальных уравнений — тоже похож на бабочку (*рис.* 2). Сколько же труда надо было приложить ученому, чтобы нарисовать эту удивительную картину?

Говоря об “эффекте бабочки”, Э.Лоренц подчеркивал, что теория хаоса связана с изучением нелинейных систем, для которых скорость изменения не является постоянной. Они характеризуются непредсказуемостью. И, следовательно, их изучение — это кропотливый и длительный процесс. Прогноз погоды и рост численности населения планеты — примеры нелинейных систем, графические модели которых строят современные компьютеры. Только им под силу сложные вычислительные задачи, связанные с изучением природы, общества, человека.

Чтобы ощутить себя в роли исследователя, можно и самому подготовить бабочку-аттрактор. Для этого надо перейти по ссылке *http://parfenov-yuriy.narod.ru/attractor.swf*. В нужных окнах ввести параметры: начальные и текущие координаты и время. Затем нажать на кнопку “Пуск” и внимательно следить за компьютером, который прямо на глазах построит замысловатую картину (*рис.* 3). Чтобы остановить процесс, надо кликнуть по надписи “Пауза”. И конечно же полюбоваться аттрактором, который нарисовала умная машина.

Рассматривая картину, нетрудно заметить, что каждая точка пространства с течением времени описывает очень сложную траекторию. Аттрактор — это множество, которое притягивает траектории движения точек. Слово “притягивает” означает, что с течением времени траектория все ближе и ближе подходит к аттрактору. По этому принципу построены аттракторы, изображенные на *рис.* 4*а*, *б*.

Рис. 2

Рис. 4*а*

Рис. 4*б*

Если траекторию немного возмутить, то есть “толкнуть” точку динамической системы в сторону, то траектория все равно будет притягиваться к множеству (*рис.* 5*а*, *б*).

Кстати, есть аттракторы, в которых траектории точек, наоборот, отталкиваются (*рис.* 6*а*, *б*). Они называются репеллерами (от англ. *to repel* — отталкивать).

Эти многочисленные и самые разнообразные “крученые” фигуры строят мощные компьютеры. Каждому аттрактору, помимо названия, соответствует своя система дифференциальных уравнений, определяющая динамическую систему. При построении указываются значения числовых параметров системы, для которых проводились расчеты. А визуализация компьютерных шедевров выполняется с помощью специальных плагинов для программного обеспечения *4D Cinema*.

Сейчас нам кажется, что компьютерные рисунки — это результат быстрых манипуляций, которые может освоить любой пользователь. Например, аттрактор, изображенный на *рис.* 7. Он напоминает маску совы или крылья бабочки. Этот рисунок стал эмблемой первых исследователей хаоса. Они пытались раскрыть тонкую структуру, таящуюся в беспорядочном потоке информации, ежедневно открывающейся человеку.

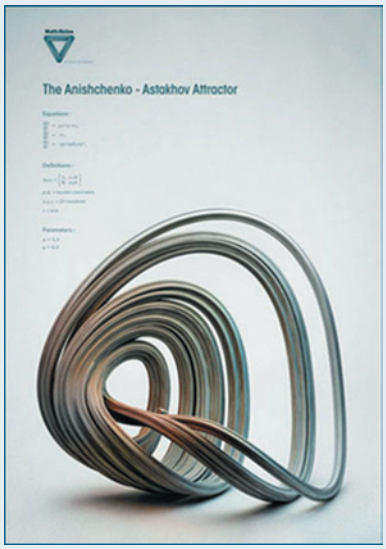


Рис. 5а

Рис. 5б

Рис. 6а

Рис. 6б

Многие годы ученые ничего не знали об аттракторах. Области знания, которыми они занимались, были изолированы друг от друга. Биологам было не до книг по математике. Физикам не хватало времени штудировать метеорологические журналы. Казалось, они были им ни к чему. На первых порах немногие оценили открытие Э.Лоренца. И еще десяток лет физики, астрономы и биологи, пытавшиеся объяснить природные явления, открывали то, что уже было открыто. Им и в голову не приходило искать описания сложных динамических систем на страницах “Метеорологического журнала”.

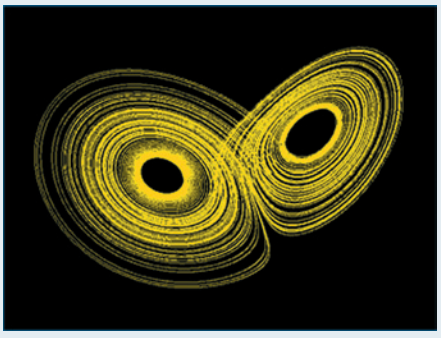


Рис. 7

Источники сети Интернет

1. *http://www.adamaz.ru/another/972-babochka-lorenca-pod-zvezdojj-vifleema.html*.
2. *http://ale.io.ua/album462682_1*.
3. *http://www.fund-intent.ru/Document/Show/3642*.
4. *http://commons.wikimedia.org/wiki/File:Atractor_Poisson_Saturne.jpg?uselang=ru*.
5. *http://www.sernam.ru/book_e_math.php?id=40*.
6. *http://www.arrays.ru/math/id/15*.
7. *http://www.runyweb.com/articles/leisure/interesting-things/gallery-of-the-most-strange-attractors.html*.
8. *http://www.ideationtriz.com/ZZLab/Nonlinear_phenomenon/Srange_attractor.htm*.
9. *http://www.polybook.ru/comma/4.8.pdf*.
10. *https://ru.wikipedia.org/wiki/%D0%90%D1%82%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BE%D1%80*.

Андроиды и электрические овцы

И.А. Сукин,
г. Переславль-Залесский

В настоящее время все большую и большую роль в нашей жизни начинают играть мобильные устройства, такие, как смартфоны, планшеты, смартбуки и другие “игрушки”, любезно предоставленные современной технологией. Разработка программ для подобных устройств долгое время оставалась достаточно сложным занятием, несмотря на существование некоего стандарта в виде J2ME. Каждое мобильное устройство обладало собственными, порой уникальными, профилями оборудования, и разработчикам приходилось проводить продолжительное время, отлаживая и оптимизируя свои программы под каждый телефон, который они хотели поддерживать в собственных приложениях.

Ситуация несколько изменилась с появлением “умных телефонов” — смартфонов и одной из первых более-менее унифицированных операционных систем для них — Symbian. Разработчики мобильных приложений в тот момент испытали вздох облегчения, поскольку работа с “внутренностями” аппаратов существенно упростилась, а интерфейсы стали более стандартизованными. Разумеется, с тех пор прошло уже немало времени, и на рынке появилось большое количество других операционных систем для мобильных устройств: Android, iOS, Bada, Tizen, Windows Phone и прочие. Одной из наиболее многообещающих и прогрессивных выглядит ОС Android (об аргументах в пользу этого — чуть позже) от Google, с достойным соперником в виде iOS для iPhone и iPad от Apple.

Зачем писать мобильные приложения?

Достаточно разумный вопрос, который может задать себе каждый читающий этот обзор: “А зачем мне вообще писать приложения для мобильных устройств?” (в предположении, что читатель уже знаком с программированием для тех же персональных компьютеров). Что могут делать мобильные приложения из того, что я не мог бы сделать на своей настольной машине?

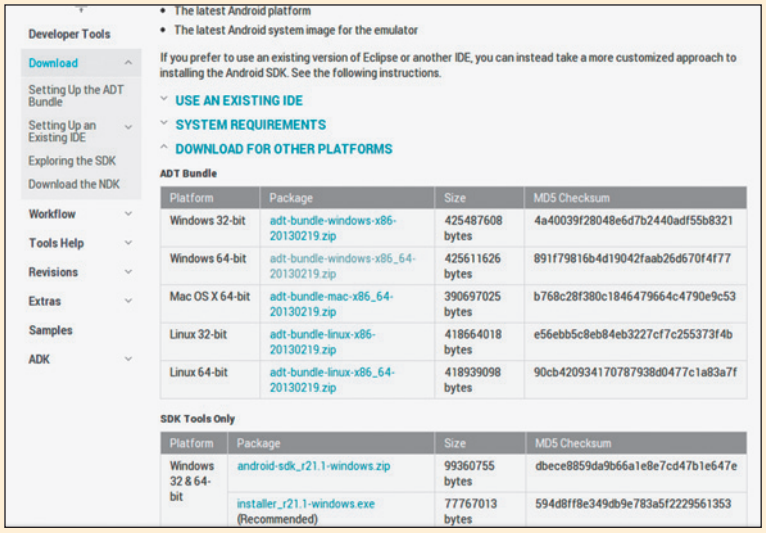
Первым и самым важным фактором здесь является, конечно, коммуникативная мобильность. Различные программы для мгновенного обмена сообщениями, документами, файлами прямо-таки просятся быть реализованными на мобильных устройствах, чтобы у человека всегда был доступ к продвинутым средствам связи с коллегами, друзьями, родственниками. Разумеется, значительное количество подобных программ уже написаны, однако не всем может нравиться конкретная реализация или может существовать какой-нибудь внутрикорпоративный протокол связи, что потребует самостоятельной реализации соответствующего приложения.

Второй важной сферой применения является область развлечений, в широком смысле, от серьезных — таких, как фотография и видеозапись (существуют даже программы для нелинейного видеомонтажа на мобильных устройствах!), до куда менее серьезных — видеоигр, которые с появлением сенсорных и мультитач-дисплеев приобрели новый особенный колорит и требуют соответствующих изменений в дизайне.

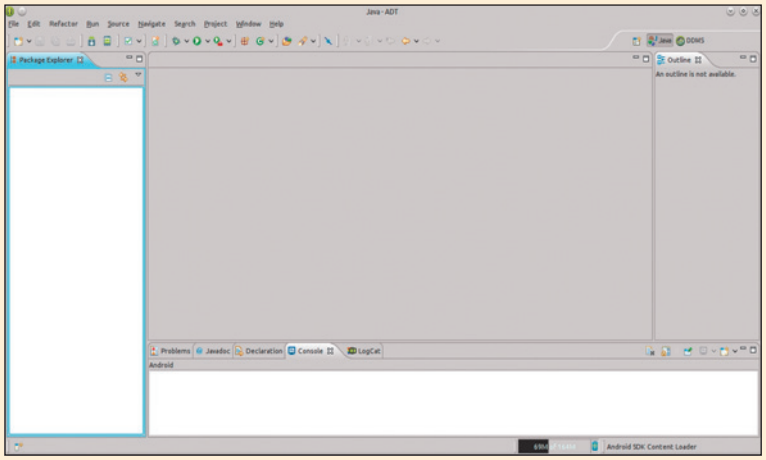
Конечно, создание таких приложений, как офисные текстовые и графические процессоры для мобильных устройств, несколько бессмысленно, в силу неудобства управления (тем не менее попытки есть). Здесь значительный интерес представляют приложения для создания и управления заметками, в особенности те, которые поддерживают рукописный ввод с распознаванием текста. В этой области пока еще есть неограниченный простор для разработки.

Кроме всего этого, серьезным преимуществом многих мобильных устройств является возможность работы с геопозиционированием, определением собственного положения через системы GPS или Глонасс и отображением этого положения и окружающей обстановки на одной из систем работы с картами.

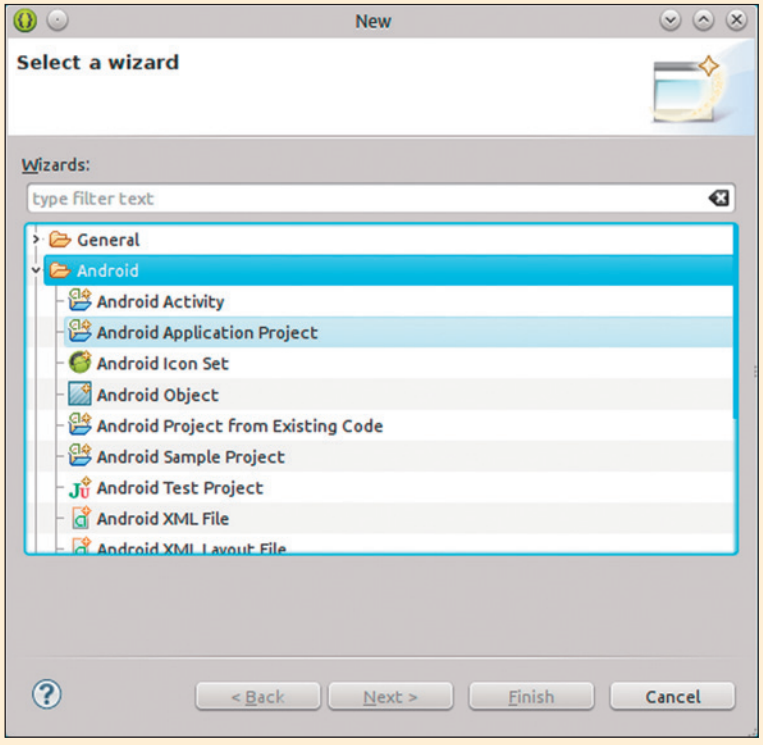
В целом можно сказать, что разработка мобильных приложений представляет собой весьма перспективную сферу, ставящую перед программистами некоторые до сих пор не возникавшие проблемы как дизайна, так и кодирования. В конце концов, это просто интересно и увлекательно.



Веб-страница со ссылками на загрузку ADT



Пустое окно Eclipse из ADT



Диалог создания нового проекта

Что такое Android?

ОС Android представляет собой операционную систему от компании Google (изначально Android разрабатывала компания Android, Inc.), долгое время существовавшую лишь на уровне слухов и выпущенную в свет лишь в 2008 году. Это по большей части открытая и свободная (исходный код опубликован под Apache License) система, основанная на ядре Linux и собственной реализации языка программирования Java — Dalvik, однако большинство производителей устройств дополнительно включает в систему закрытые проприетарные модули для поддержки собственного оборудования. Сегодня Android можно встретить на большом спектре устройств: от мобильных телефонов до планшетов, электронных книг и даже ноутбуков.

Многие архитектурные решения и возможности обеспечили Android быстрый рост числа пользователей и одну из лидирующих позиций на рынке. Одной из таких возможностей была, в частности, качественная и унифицированная графика на основе OpenGL.

Более подробно на некоторых возможностях Android я остановлюсь позднее.

Web-приложения или мобильные приложения?

Еще одним значительным веянием современной технологии стали web-приложения, работающие непосредственно в браузере и не требующие какой-либо поддержки со стороны операционной системы. Многие разработчики, в том числе и мобильных программ, избрали для себя этот подход. В чем его преимущества? Во-первых, браузеры есть практически везде, практически на любом устройстве: от самых примитивных мобильных телефонов до персональных компьютеров и мейнфреймов. Во-вторых, программирование web-приложений осуществляется безотносительно какой-либо определенной аппаратной платформы — браузер предоставляет значительную степень кроссплатформенности, которая ограничивается только некоторой несовместимостью различных браузерных движков (количество основных из них в последнее время сократилось до трех: WebKit — Chrome, Opera, Safari, Gecko — Firefox, Trident — Internet Explorer). В-третьих, для web-приложения отсутствует понятие “обновления” — программа обновляется автором на собственном сервере, что никак не затрагивает конечных пользователей. И, наконец, что также немаловажно, существует большое количество зрелых и устоявшихся технологий web-программирования: PHP, Perl, Python/Django, Ruby/Rails и другие, с обширной литературой по большинству вопросов и огромным количеством стандартных методов решения типовых программистских задач.

Казалось бы, с web-программированием все замечательно, и мы должны избрать его в качестве единственного приемлемого варианта для разработки наших мобильных приложений. Но у этого подхода есть существенные минусы, которые сильно ограничивают нашу сферу. Во-первых, и в главных, web-приложения не имеют никакого доступа к нижележащей операционной системе, будь то доступ к файлам или многим устройствам (за исключением, пожалуй, web-камеры). Такой доступ существует только в Java-апплетах или приложениях на ActiveX, Flash, Silverlight, при этом теряется значительная доля кроссплатформенности, и, кроме этого, такие обходные пути считаются дурным тоном. Вторым серьезным недостатком является слишком ограниченный набор графических компонентов для создания интерфейсов пользователя без какой-либо возможности серьезного комбинирования компонентов и создания собственных виджетов; отсутствует прямая отрисовка (частично эта ситуация исправлена с появлением HTML5 Canvas и WebGL). Способ решения здесь такой же, как и в предыдущем случае, то есть неприемлемый для нас. В качестве более мелких, но довольно назойливых недостатков можно назвать изначальную ориентированность web-технологий на устройства с большим дисплеем и мышью, а не на сенсорные и небольшие экраны, а также несовершенность, как с точки зрения оптимальной реализации, так и с позиции безопасности, протокола HTTP.

Мобильные приложения, написанные под конкретную операционную систему, лишены вышеперечисленных недостатков, они, как правило, поддерживают большое число графических виджетов с возможностью их комбинирования и создания новых, пользовательских, имеют доступ к прямой отрисовке через OpenGL, доступ к файловой системе, базам данных, устройствам вроде микрофона, акселерометра, сканера отпечатков пальцев, могут совершать телефонные вызовы и отправлять сообщения. Виджеты при этом оптимизированы для удобства использования на экранах мобильных устройств, а для коммуникации программист волен использовать любые протоколы, какие ему вздумается. Технологии разработки мобильных приложений, такие, как Java (Android) и Objective C (iOS), тоже имеют за собой большую базу.

К несомненным минусам чисто мобильных приложений следует отнести низкую кроссплатформенность, к примеру, вы не сможете запустить приложение для Android на iPhone, и необходимость непосредственного участия пользователя в процессе обновления программного обеспечения.

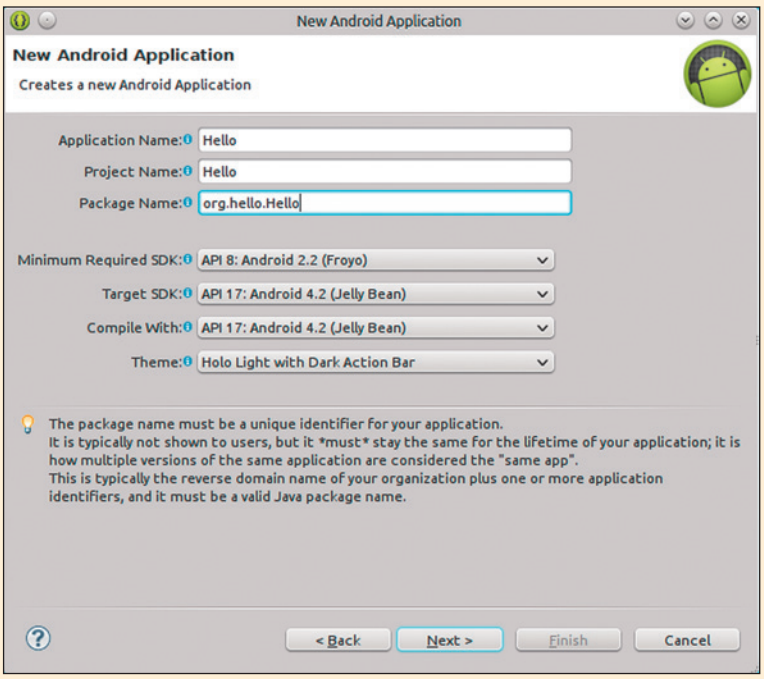
Резюмируя вышесказанное, можно сделать вывод, что web-приложения превосходно подходят для решения задач, не требующих обращения к файлам и устройствам, а также имеющих простой интерфейс, то есть типичных программ вроде web-интерфейсов к информационным системам и некоторых простейших развлекательных приложений, например, игр. Для всех остальных проблем разумнее писать приложения “в железе”.

Почему Android, а не iOS?

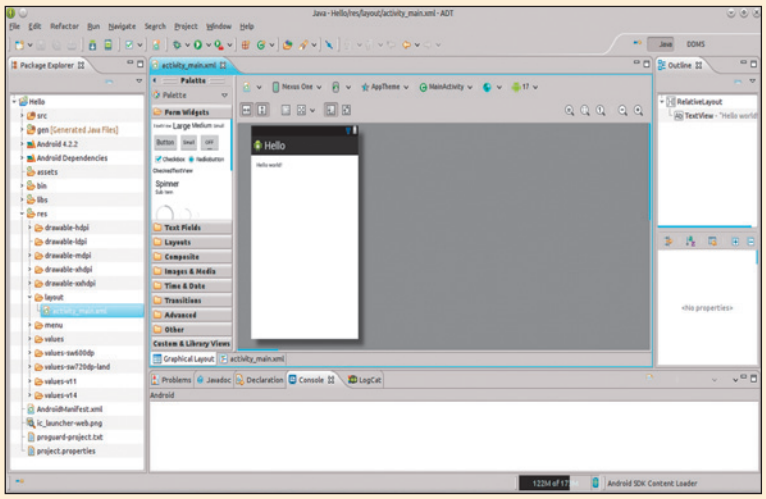
Если рассматривать статистику продаж мобильных устройств, в особенности телефонов, пока что лидирующие позиции занимает продукция фирмы Apple — iPhone и iPad. Стоит ли при написании мобильных приложений ориентироваться на них и какие здесь есть преимущества у Android? Проведем небольшое сравнение.

Начнем с простых вещей, актуальных не только для разработчиков: способа загрузки приложений на устройство. Для iPhone и iPad существует только один легальный способ сделать это — послать вашу программу в Apple App Store и дождаться возможного (но не гарантированного) ее одобрения со стороны Apple. К сожалению, это необходимо проделывать даже с вашими личными приложениями, вроде маленьких утилит. Другой способ — сделать так называемый “jailbreak” вашего устройства, что, однако, запрещено законодательством многих стран. Чем здесь отвечает Android? Во-первых и в главных, вы можете загрузить вашу программу непосредственно на телефон через USB-кабель, что, признайтесь, действительно очень удобно. Во-вторых, несмотря на то что существует “стандартный” магазин приложений Google App Store, есть также несколько независимых от него, и в том числе неограничены возможности собственного создания таких порталов.

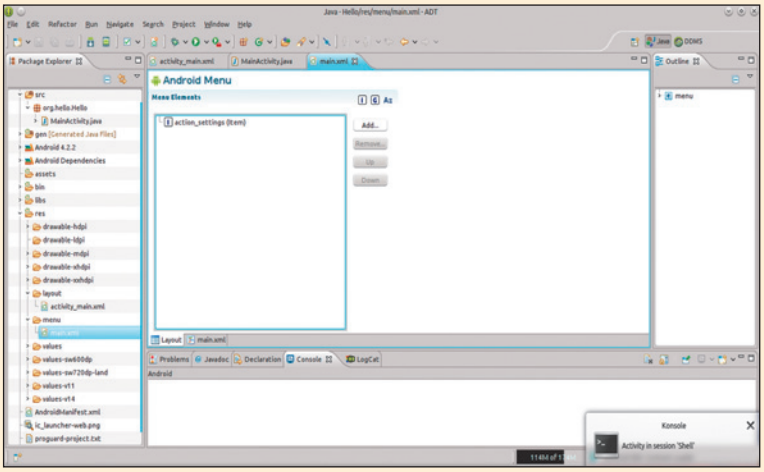
Обратимся теперь к технической стороне вопроса, а именно к языкам, средам и инструментам программирования. Программирование для



Начальный диалог с параметрами нового проекта для Android



Свежесозданный проект с открытым визуальным редактором “главной активности”



Java-код логики “главной активности”

iOS осуществляется на языке Objective C, который привычен только для тех, кто уже имеет опыт разработки для MacOS. При этом разумеется, что вся разработка производится только на Mac'ах в среде Xcode. Программировать для Android же можно под всеми основными OS, на языке Java, который в настоящее время является одним из самых популярных в корпоративном секторе, в средах Netbeans или Eclipse.

Обобщив, можно сказать, что для разработки личных, внутрикорпоративных или открытых программ Android подходит как нельзя лучше. Программирование под iOS же имеет смысл, если вы хотите продавать ваше приложение (однако динамика рынка в настоящее время показывает рост доли устройств на Android).

Что нужно для программирования под Android?

Для того чтобы разрабатывать собственные приложения для платформы Android, в первую очередь необходимо знать язык программирования Java. Дальше в тексте я буду предполагать, что читатель знает его хотя бы на начальном уровне и может читать простейшие программы на нем.

Во-вторых, как я уже говорил выше, потребуется среда разработки, такая, как Eclipse или Netbeans, а также Android SDK — официальный набор разработчика. Я буду использовать стандартный SDK, предлагаемый Google, и который можно скачать с сайта *http://developer.android.com/sdk*. В состав этого набора по умолчанию входит среда Eclipse. Разумеется, некоторый опыт работы с ней тоже будет полезным.

Установка необходимых компонентов

Для того чтобы начать программировать под Android, вам необходимо скачать Android SDK для вашей процессорной архитектуры и операционной системы с веб-страницы *http://developer.android.com/sdk*. Как я уже говорил выше, одной из основных частей этого комплекта является среда разработки Eclipse. Конечно, для любителей командной строки существуют все необходимые утилиты, однако для простоты я буду пользоваться IDE.

Начальная настройка в данном случае предельно проста: просто распакуйте скачанный архив с ADT (***A**ndroid **D**evelopment **T**ools*) в любое удобное вам место. Назовем это место $ADT_HOME. В директории $ADT_HOME/eclipse будет находиться исполняемый файл Eclipse, являющийся для нас основным.

Теперь необходимо будет создать виртуальное Android-устройство, если вы не хотите каждую новую тестовую версию своей программы загружать на реальный аппарат. Чтобы сделать это, нужно запустить Eclipse из ADT, зайти в меню Window → Android Virtual Device Manager и в появившемся окне нажать кнопку New. После этого появится довольно большой и подробный диалог с выбором характеристик создаваемого устройства. Нас в данный момент будут интересовать лишь три поля: AVD Name, Device и Target. Текстовое поле AVD Name содержит имя создаваемого виртуального устройства, например, "test_device". Выпадающий список Device позволяет выбрать один из предустановленных наборов настроек, соответствующих реальным устройствам. При разработке с оптимизацией под конкретную модель лучше выбрать что-нибудь похожее, в данный же момент можно выбрать первую попавшуюся конфигурацию. Пусть это будет "Nexus 7". Выпадающий список Target используется для выбора программной платформы, и на момент написания статьи в нем присутствует только один вариант — Android 4.2.2 с 17-й версией API. После установки всех необходимых параметров необходимо нажать кнопку Ok для завершения создания виртуального устройства.

Ваше первое приложение

Будем предполагать, что читатель установил и настроил ADT и запустил входящую в его комплект среду Eclipse. Для создания проекта Android-приложения необходимо нажать на иконку New в панели инструментов (можно, конечно, и воспользоваться соответствующим пунктом меню). В появившемся после этого диалоге, в папке Android нужно выбрать элемент Android Application Project и нажать Next.

Открывающийся после этого диалог позволяет выбрать имя вашей программы, название проекта и название пакета, в котором будет располагаться исходный код. Выпадающие списки, расположенные ниже, служат для выбора минимальных версий платформы и SDK, необходимых для запуска и компиляции программы, а также ее графической темы. Наберем в качестве имени приложения "Hello" (имя проекта при этом автоматически станет таким же), а в качестве названия пакета — org.hello.Hello; выпадающие списки оставим нетронутыми; после этого нажмем Next.

Все следующие диалоги пока что можно безболезненно пропустить, нажимая Next до тех пор, пока не появится возможность нажать кнопку Finish, после чего проект будет создан и открыт в Eclipse. Слева в виде дерева будет располагаться привычное для большинства IDE дерева проекта, а в центре — визуальный редактор, в котором можно поместить на некоторой "форме" компоненты пользовательского интерфейса. Пока что приложение содержит одну-единственную текстовую строку "Hello world!".

Для запуска приложения (компиляция по умолчанию происходит автоматически) необходимо щелкнуть на стрелочку рядом с кнопкой Run в панели инструментов и выбрать пункт, соответствующий вашей программе (в нашем случае Hello). После этого приложение будет загружено на эмулятор (виртуальное устройство) и запущено на нем.

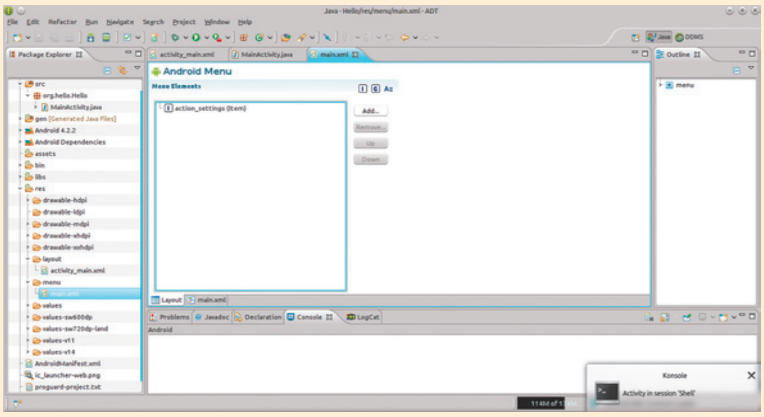
Кроме запуска на эмуляторе, приложение можно запустить на реальном устройстве. Для этого нужно, установив необходимые драйверы (процесс может отличаться для каждого устройства), присоединить устройство к компьютеру с помощью USB-кабеля и включить в настройках “Отладку по USB”. В версиях Android 3.2 и раньше необходимый флажок находится в меню Настройки → Приложения → Разработка, в Android 4.0–4.1 в меню Настройки → Опции разработчика. В самых новых версиях Android, начиная с 4.2, необходимо совершить дополнительные действия для включения отладки по USB. Точный способ включения этой возможности для конкретного устройства должен быть описан в руководстве по его эксплуатации. Теперь при попытке запуска программы способом, описанным чуть ранее, появится диалог, позволяющий выбрать либо реальное, либо виртуальное устройство.

Основные принципы программной среды Android

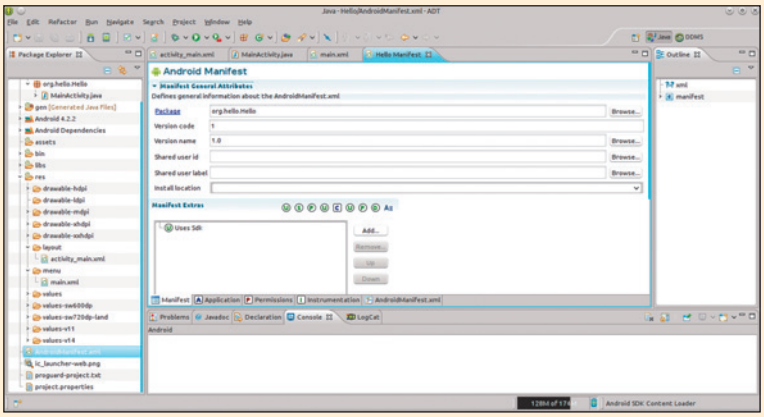
Как я уже говорил ранее, приложения для Android представляют собой программы на языке Java. Эти программы упаковываются вместе со всеми данными и ресурсами в один apk-файл, который и является главным файлом приложения.

На каждом устройстве Android-приложение существует в собственной программной среде, для обеспечения безопасности. Основные принципы здесь таковы:

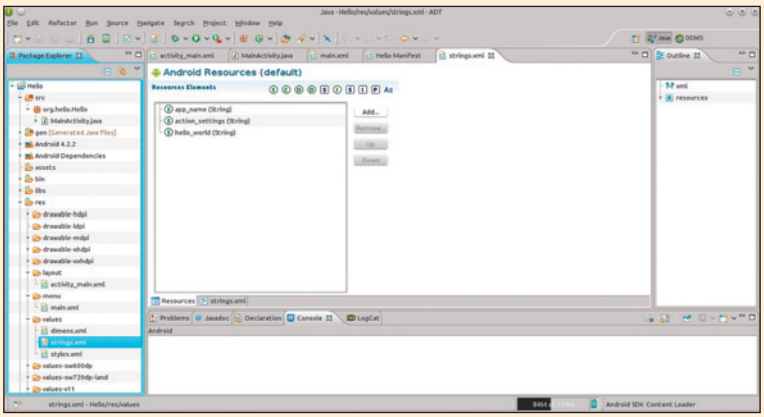
- ОС Android — это, в своей основе, многопользовательская операционная система, основанная на Linux, при этом для каждого приложения создается отдельный пользователь;
- Система устанавливает такие права на доступ к файлам приложения, что только оно само может получить к ним доступ;



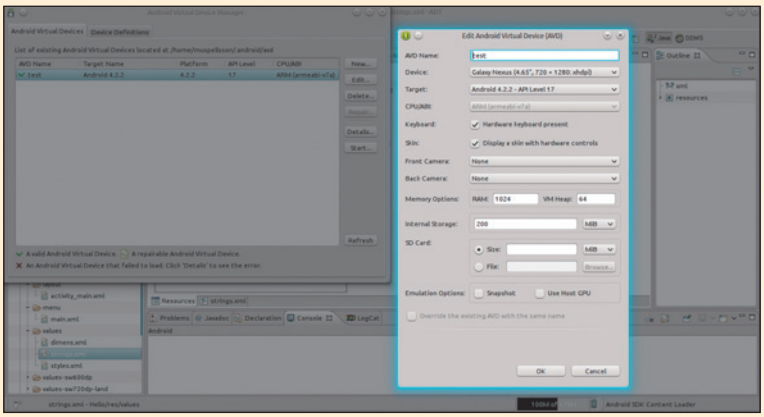
Редактор меню



Редактор главного манифеста



Редактор строковых ресурсов



Диалог создания нового виртуального устройства

• Каждая программа запускается в своей собственной виртуальной машине, таким образом, программный код одного приложения не имеет представления о коде другого;

• Разумеется, каждое приложение запускается в рамках своего собственного процесса ОС Linux, что также обеспечивает дополнительную безопасность.

Подобный подход называется *принципом наименьших привилегий*, то есть приложение обладает минимально возможными правами, необходимыми для запуска и работы.

Существуют, однако, способы получения дополнительных полномочий для программы:

• Два приложения могут сосуществовать под одним идентификатором пользователя, тем самым имея доступ к файлам друг друга. Можно также запустить программы в рамках одного процесса и одной виртуальной машины (однако для этого они должны быть подписаны одним и тем же сертификатом безопасности);

• Приложение может запросить доступ к дополнительным ресурсам, таким, как список контактов, отправка сообщений, камера, внешняя карта памяти и т.п. Разрешение на такой доступ дает пользователь на этапе установки приложения.

Компоненты приложений

Приложения для Android строятся из так называемых "компонентов", которые являются средствами взаимодействия между программой и системой. Всего существует четыре различных вида компонентов:

• ***Активность (Activity)*** представляет собой попросту пользовательский интерфейс. Данный компонент можно воспринимать как аналог "формы" в таких средах, как Delphi. Приложение может иметь несколько активностей, при этом активность, отображаемая первой, считается главной. Более того, приложение может отображать активности других приложений, что для конечного пользователя выглядит гладко, то есть так, будто вся работа происходит в рамках одной программы. С этой точки зрения активность можно рассматривать как некое подобие веб-страницы.

• ***Сервис*** — это компонент, работающий в фоновом режиме и обеспечивающий выполнение продолжительных задач, не требующих прямого вмешательства пользователя. К примеру, сервис может проигрывать музыку или передавать данные по сети.

• ***Контент-провайдер (Content Provider)*** служит для организации доступа к данным, которые должны быть общими для нескольких приложений, таким, как файлы или базы данных. Ярким примером контент-провайдера является список контактов.

• ***Broadcast Receiver*** используется для реагирования на общесистемные события и уведомления, такие, как сигнал о выключении экрана, уровне заряда батареи, входящем сообщении, съемке фотографии.

Все компоненты могут быть глубоко связаны друг с другом с помощью программной логики.

Каждый компонент перед использованием должен быть *активирован*. Происходит это с помощью посылки специального сообщения-*намерения*. С помощью такого сообщения можно активировать практически любой компонент любого приложения в нужное время, что фактически означает, что приложения не имеют единой точки входа (такой, как функция main()). Это обеспечивает достаточную гибкость работы вкупе с высокой безопасностью и степенью разделения ресурсов. Так как одна программа не имеет непосредственного доступа к компонентам другой, такие действия, как запуск *активности* одного приложения из другого, реализуются путем сообщения системе о своем *намерении* совершить это.

Файл манифеста

Одной из основных составляющих частей любого Android-приложения является файл AndroidManifest.xml, содержащий всю основную информацию о программе. В частности, в нем описываются:

• желаемые привилегии приложения и доступ к дополнительным устройствам и хранилищам данных;

• минимальная версия API, необходимая для запуска и работы приложения;

• декларации используемых программных и аппаратных возможностей;

• дополнительные используемые библиотеки, такие, например, как Google Maps API;

• объявления компонентов, их возможностей и требований (например, минимально необходимого разрешения экрана для *активности*).

В среде Eclipse файл AndroidManifest.xml можно редактировать не только в виде XML-текста, но и с помощью более удобных диалогов, в чем можно убедиться, дважды кликнув на этом файле в дереве проекта.

Ресурсы

Конечно, редко какое приложение состоит исключительно из программного кода, большая часть полезных программ имеют в своем составе файлы с данными. Такие файлы в контексте Android именуются *ресурсами*. Для каждого *ресурса* Android SDK создает некоторый идентификатор, который потом можно использовать для обращения к *ресурсу* в коде. К примеру, для файла image.png, лежащего в директории res/drawable, будет создан идентификатор "R.drawable.image".

Важной особенностью *ресурсов* в Android является возможность использования в рамках одного приложения разных *ресурсов* на разных устройствах или разных конфигурациях оборудования (например, загрузка других изображений при изменении ориентации экрана с портретной на альбомную). Поддержка этого обеспечивается с помощью *квалификаторов ресурсов*. Здесь я не буду касаться этой темы подробно.

Разбор нашего первого приложения

Теперь, получив представление об основных принципах программного окружения Android, можно заняться непосредственным разбором кода нашего приложения Hello.

Обратимся к дереву проекта. Первым каталогом в этом дереве является "src", где хранятся все исходные файлы на языке Java. В настоящий момент там в пакете org.hello.Hello лежит файл MainActivity.java, отвечающий за логику работы *главной активности* нашего приложения, той самой, в которой отображается текст "Hello world!". Если посмотреть на исходный код, то можно увидеть, что создается новый класс MainActivity, расширяющий стандартный класс Activity, и переопределяются два метода: onCreate и onCreateOptionsMenu. Первый метод, onCreate, приведен ниже.

```
protected void onCreate(Bundle
                        savedInstanceState)
{ super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
}
```

Первым делом в нем вызывается соответствующий метод суперкласса, после чего устанавливается содержимое *активности* из некоторого *ресурса*. Здесь можно заметить автоматически сгенерированный идентификатор "R.layout.activity_main", указывающий на файл res/layout/activity_main.xml (к этому файлу мы еще вернемся чуть позже), содержащий описание пользовательского интерфейса. Метод onCreate — единственный метод, который наш свеженаписанный класс MainActivity должен переопределить.

Второй метод, onCreateOptionsMenu:

```
public boolean onCreateOptionsMenu(Menu menu)
{
  //Inflate the menu; this adds items
    to the action bar if it is present.
    etMenuInflater().inflate(R.menu.main, menu);
    return true;
}
```

Этот метод служит для создания стандартного меню настроек приложения. Как можно увидеть из кода, описание меню берется из файла res/menu/main.xml (к нему мы тоже вернемся позже). Я не буду подробно останавливаться на этом методе.

Второй каталог в дереве проекта, "gen" содержит файлы кода на Java, сгенерированные автоматически. Для нас они не представляют практически никакого интереса. В данный момент там находятся два файла: BuildOptions.java, содержащий настройки текущего режима компиляции, и R.java, содержащий автоматические определения идентификаторов ресурсов.

Каталоги "Android 4.2.2" и "Android Dependencies" содержат jar-архивы всех стандартных библиотек платформы Android.

Каталог "assets" служит для хранения файлов. Он отличается от каталога ресурсов тем, что файлы там являются "динамическими", то есть не требуют объявления во время компиляции. При этом, конечно, теряются все описанные ранее преимущества, которыми обладают *ресурсы*, как то автоматически созданные идентификаторы и автоматический выбор ресурсов для разных конфигураций.

В каталоге "bin" находятся важные исполняемые файлы, необходимые для работы приложения, в частности, наш основной файл "Hello.apk".

Каталог "libs" содержит все необходимые дополнительные библиотеки.

Каталог "res", как читатель уже мог понять, содержит *ресурсы*, разбитые по категориям. Например, в данный момент там есть пять подкаталогов: drawable-hdpi, drawable-ldpi, drawable-mdpi, drawable-xhdpi, drawable-xxhdpi, — в которых содержатся файлы изображений (конкретно сейчас — иконки) для различных вариантов разрешения экрана.

В подкаталоге "res/layout" лежат XML-файлы, описывающие интерфейс пользователя, то есть *активности*. Рассмотрим файл main_activity.xml, содержащий описание *главной активности* нашего приложения:

```
<RelativeLayout
    xmlns:android=
  "http://schemas.android.com/apk/res/android"
    xmlns:tools=
            "http://schemas.android.com/tools"
    android:layout_width=
                          "match_parent"
    android:layout_height=
                          "match_parent"
    android:paddingBottom=
             "@dimen/activity_vertical_margin"
    android:paddingLeft=
           "@dimen/activity_horizontal_margin"
    android:paddingRight=
           "@dimen/activity_horizontal_margin"
    android:paddingTop=
             "@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</RelativeLayout>
```

Его структура проста. Кроме тега RelativeLayout со служебными атрибутами, он содержит один-единственный тег TextView, описывающий нашу надпись “Hello world!”. Запись “@string/hello_world” является ссылкой на строковый ресурс. Более подробно такой тип ресурсов мы рассмотрим позже. Eclipse также позволяет редактировать пользовательский интерфейс визуально, такой подход хорошо знаком всем тем, кто программировал в визуальных средах вроде Delphi. К примеру, если мы произвольным образом поместим на “форму” кнопку, то в файл main_activity.xml будут добавлены строчки вроде

```
<Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView1"
        android:layout_marginLeft="59dp"
        android:layout_marginTop="152dp"
        android:layout_toRightOf=
                              "@+id/textView1"
        android:text="Button" />
```

описывающие только что добавленный виджет. Здесь можно также обратить внимание на то, что новые компоненты можно выравнивать относительно уже существующих. Создавать ли интерфейс с помощью графических средств или же редактируя текстовые файлы, зависит от ваших личных предпочтений, однако, графический дизайнер предоставляет очень большую гибкость.

Немножко отвлекаясь от основной темы раздела уделю время описанию того, как заставить кнопку реагировать на нажатие. Для описания этого действия применяется атрибут android:onClick тега Button, значением которого является имя метода в классе, соответствующем описываемой *активности*. Этот метод должен быть объявлен как public void и принимать один аргумент типа View. Например:

```
<Button
     android:id="@+id/button1"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:layout_below="@+id/textView1"
     android:layout_marginLeft="59dp"
     android:layout_marginTop="152dp"
     android:layout_toRightOf="@+id/textView1"
     android:onClick="destroyUniverse"
android:text="Button" />
```

где метод destroyUniverse будет иметь прототип

```
public void destroyUniverse(View view)
```

Каталог “res/menu” содержит XML-файлы с описанием меню. Как и в случае с дизайном *активностей*, меню можно редактировать как с помощью графического диалога, так и с помощью прямой работы с XML. Здесь я не буду подробно рассматривать создание меню.

Каталог “res/values” служит для хранения XML-файлов с всевозможными текстовыми ресурсами. В данный момент в каталоге находятся три файла. Первый из них, dimens.xml, хранит параметры экрана, второй, strings.xml, используется для хранения строковых ресурсов, таких, как уже встретившаяся нам ранее надпись “Hello world!”. Третий файл, styles.xml, содержит информацию о теме графического оформления приложения. Все эти файлы можно редактировать как в диалоговом режиме, так и в текстовом.

Следующие несколько подкаталогов в “res”:
values-sw600dp,
values-sw720dp-land,
values-v11,
values-v14
— содержат описания разрешений экрана и тем для различных программно-аппаратных конфигураций.

Еще один важный файл в дереве проекта — AndroidManifest.xml мы уже рассмотрели выше. Все остальные файлы служат для всевозможных вспомогательных целей, являются автоматически сгенерированными и в данный момент не представляют никакого интереса.

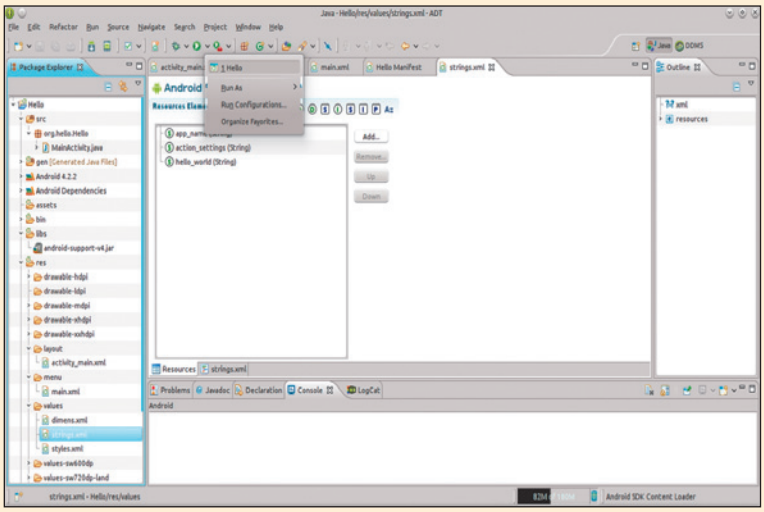
Что дальше?

Создание полноценных приложений для Android — сложная и чрезвычайно обширная область технологии, по которой написаны объемные книги, и в рамках одной статьи сложно дать даже беглый обзор всех предоставляемых возможностей и методов. В стандартную поставку Android SDK входит набор примеров, по которым можно учиться писать более сложные приложения, а на официальном сайте *http://developer.android.com/training/* есть большое количество учебного материала.

Другие способы разработки для Android

ОС Android пока еще слишком молода, а ее инфраструктура до сих пор менялась слишком быстро, поэтому в настоящий момент существует слишком малое число альтернативных способов разработки приложений. В частности, существуют некоторые попытки перенести интерпретаторы скриптовых языков на Android, самой значительной из которых является Script Layer for Android (SL4A). Также существуют работающие прототипы систем программирования для языков Smalltalk и J.

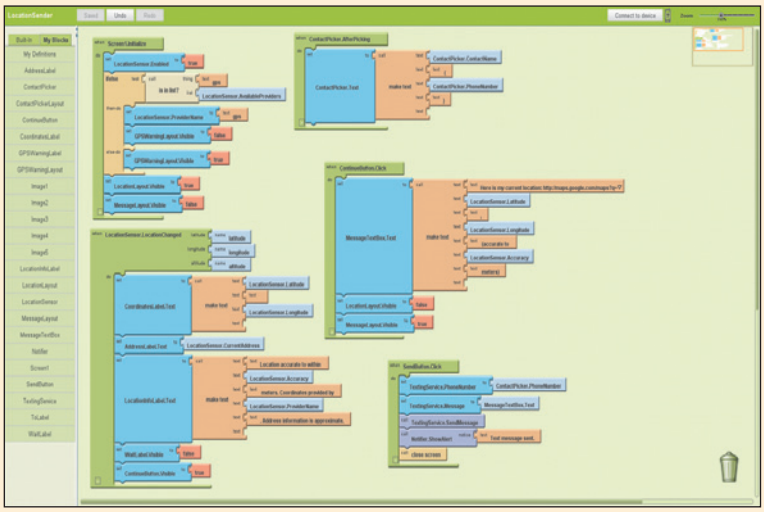
Одним из наиболее значительных альтернативных способов разработки программ для Android является MIT App Inventor — визуальная среда разработки, похожая на Scratch (среду для обучения программированию), в которой для создания приложений используется компоновка графических блоков. Приложения создаются в браузере на компьютере разработчика, после чего либо загружаются на устройство по USB, либо просто сохраняются в виде apk-файла. App Inventor пока находится в стадии бета-тестирования.



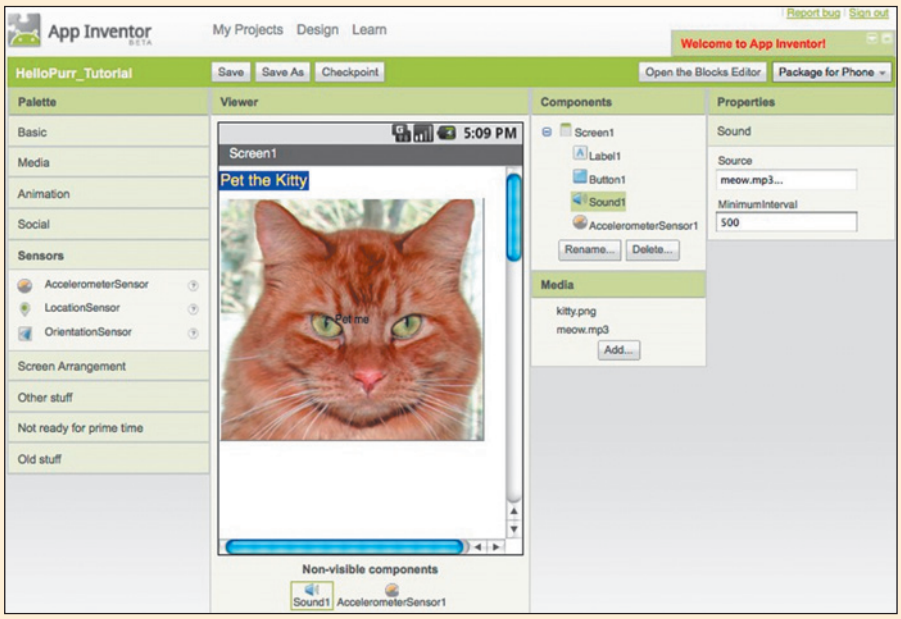
Меню запуска приложения

Список литературы

1. *Ed Burnette*. “Hello, Android”, The Pragmatic Programmers, 2010.
2. Официальная документация к Android. *http://developer.android.com/reference*.
3. Учебные материалы по программированию для платформы Android. *http://developer.android.com/training*.
4. *W. Frank Ableson, Charley Collins and Roby Sen*. Unlocking Android. A Developers Guide, Manning Publications Co., 2009.
5. *Голощапов Алексей*. Google Android. Программирование для мобильных устройств. BHV-СПб., 2010.
6. Документация к MIT App Inventor. *http://appinventor.mit.edu/explore/*.



Редактирование логики в App Inventor



Редактирование визуальных элементов в App Inventor

Педагогический университет
«Первое сентября»

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

ДИСТАНЦИОННЫЕ КУРСЫ ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

(с учетом требований ФГОС)

Ведется прием заявок на 2013/14 учебный год

образовательные программы:

- НОРМАТИВНЫЙ СРОК ОСВОЕНИЯ **108** УЧЕБНЫХ ЧАСОВ

  Стоимость – 2990 руб.

- НОРМАТИВНЫЙ СРОК ОСВОЕНИЯ **72** УЧЕБНЫХ ЧАСА

  Стоимость – 2390 руб.

По окончании выдается удостоверение о повышении квалификации установленного образца

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

Пожалуйста, обратите внимание:

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

журнал

Информатика – Первое сентября

ТАРИФНЫЕ ПЛАНЫ НА ПОДПИСКУ

2-е полугодие 2013 года

Максимальный – 1440 руб.

бумажная версия (по почте) + CD + доступ к электронной версии на сайте

Оформление подписки – на сайте www.1september.ru или на почте по каталогам:
«Роспечать» – индекс 32291 *(для индивидуальных подписчиков и организаций)*
«Почта России» – индекс 79066 *(для индивидуальных подписчиков и организаций)*

Оптимальный – 594 руб.

электронная версия на CD (по почте) + доступ к электронной версии на сайте

Оформление подписки – на сайте www.1september.ru или на почте по каталогам:
«Роспечать» – индекс 19179 *(для индивидуальных подписчиков и организаций)*
«Почта России» – индекс 12684 *(для индивидуальных подписчиков и организаций)*

Экономичный – 300 руб.

доступ к электронной версии и оформление подписки на сайте www.1september.ru

Бесплатный – 0 руб.

доступ к электронной версии на сайте www.1september.ru для педагогических работников образовательных учреждений, участвующих в Общероссийском проекте «Школа цифрового века»

Бумажная версия (доставка по почте)

CD с электронной версией журнала и дополнительными материалами для практической работы (доставка по почте)

Электронная версия в Личном кабинете подписчика на сайте www.1september.ru Дополнительные материалы включены

Пользователям электронной версии высылаются по почте подтверждающие документы

ЭКОНОМИЧНЫЙ тарифный план

ОПТИМАЛЬНЫЙ тарифный план

МАКСИМАЛЬНЫЙ тарифный план

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

VISA MasterCard QIWI WebMoney

ГРАФИКА

“Занимательный практикум”: изучаем графический редактор

О.Б. Богомолова,
преподаватель Дворца творчества детей и молодежи “Преображенский”, Москва
Д.Ю. Усенков, ст. науч. сотр. Института информатизации образования РАО, Москва

Предлагаем подборку подобных занимательных практических заданий, которые позволят вам освоить ряд операций по работе с компьютерной графикой — выделение, копирование (вырезание) и вставку фрагментов, работу со слоями, использование таких инструментов, как “Лупа” и “Ластик”, и др. Можно сказать, что задания представляют собой увлекательную игру по созданию необычных фотоколлажей и художественных фоторамок, которые пригодятся в повседневной жизни, например, для оформления личного фотоальбома.

В качестве программной базы для выполнения этих заданий выбран графический редактор Paint.Net. Его преимущества:

- бесплатность;
- простота освоения (по сравнению, например, с профессиональным редактором Adobe Photoshop, насыщенным большим количеством необязательных функций, затрудняющих его освоение для начинающих);
- достаточно богатый набор предоставляемых возможностей, гораздо больший, чем у простейших растровых редакторов типа Paint (по возможностям Paint.Net лишь незначительно уступает тому же Adobe Photoshop).

Дистрибутив программы Paint.Net имеется на диске к данному выпуску “Информатики”, а прочитать подробную инструкцию к ней на русском языке можно на сайте *http://paintnet.ru/?id=2*.

1. Фотомонтаж: размещение фотографии в художественной рамке (см. *рис*. 5)

Откройте программу Paint.Net, щелкнув мышью на ее пиктограмме:

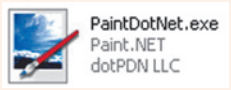


Рис. 1

Откроется диалоговое окно программы:

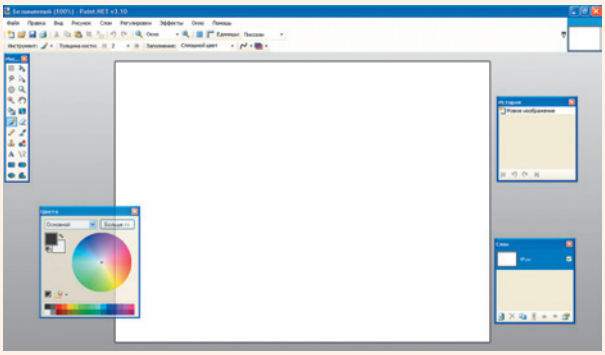


Рис. 2

Чтобы поместить свою фотографию в готовую художественную рамку, используя готовый шаблон:

- откройте шаблон *Рамка.jpg*[1] (меню **Файл — Открыть**);
- выберите инструмент “Волшебная палочка”  и щелкните мышью в шаблоне — фоторамке на том участке, где должна быть размещена фотография (на одноцветном закрашенном поле);
- если при выделении этой одноцветной области выделение захватывает другие части изображения фоторамки или если, наоборот, по краям остались не выделенные части поля для фотографии, то настройте точность выделения при помощи “волшебной палочки” с помощью ее вспомогательного инструмента “Точность” 

. Для этого щелкните левой кнопкой мыши в соответствующем месте полоски — она будет закрашиваться слева и до места щелчка, и на ней будет меняться значение в

[1] Все графические файлы, необходимые для выполнения этого и других заданий, представлены на диске к данному номеру журнала.

процентах. Затем повторно попытайтесь выделить область при помощи “волшебной палочки”. В результате должна быть выделена именно та часть фоторамки, в которой должна располагаться ваша фотография;

• нажмите клавишу **Delete** (или **Del**) на клавиатуре. Выбранная область станет прозрачной — в графическом редакторе она выглядит как шахматное поле из белых и серых квадратиков;

• откройте фотографию *Конкурс.jpg*[2] (меню **Файл** | **Открыть**);

• выделите нужную часть фотографии инструментом выделения и скопируйте фотографию в буфер обмена (меню **Правка** | **Копировать**);

• перейдите к фоторамке и выполните команду меню **Правка** | **Вставить в новый слой**. При необходимости выберите в окне запроса пункт `Expand canvas` (“Увеличить холст” — поле, занимаемое рисунком) — *рис*. 3;

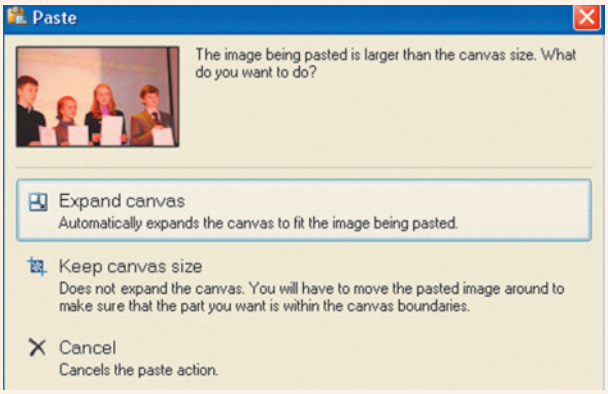


Рис. 3

• используя кнопку в окне **Слои**, переместите слой с вставленной фотографией вниз, под слой фоторамки;

• проверьте, чтобы в окне **Слои** был выделен слой с фотографией (для выделения слоя достаточно щелкнуть на его имени мышью);

• подгоните фотографию по размерам и положению так, чтобы она просвечивала сквозь прозрачную область фоторамки;

• закончив работу, сохраните получившуюся фотографию в рамке с помощью команд **Файл** | **Сохранить как**. Задайте имя файла *Фото в рамке*, а расширение файла — *png*. Нажмите кнопку **Сохранить**;

• появится диалоговое окно **Слить слои?** (*рис*. 4). Выберите в нем `Flatten` (“объединение”);

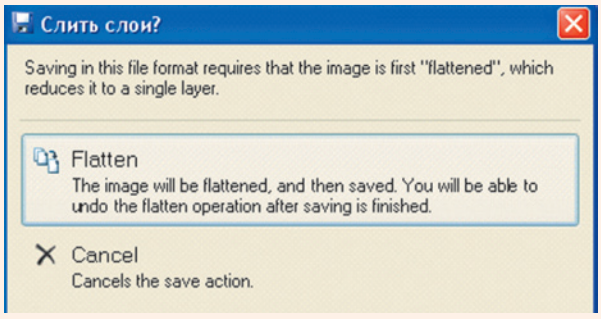


Рис. 4

[2] Можно также использовать собственные фотографии. — **Прим. ред.**

• если при вставке фотографии вы выбирали `Expand canvas`, выделите фото в рамке с помощью инструмента “Прямоугольное выделение” и обрежьте белые поля с помощью инструмента “Обрезать по выделенному” ;

• сохраните результат (меню **Файл** | **Сохранить**). Фото в рамке готово!

Рис. 5

2. Фотомонтаж: размещение вашего лица в готовом коллажном шаблоне (см. *рис*. 7)

Откройте программу Paint.Net, щелкнув мышью на ее пиктограмме, — появится диалоговое окно программы (см. *рис.* 2).

Чтобы поместить фрагмент фотографии (лицо) в готовый коллажный шаблон:

• откройте файл *Готовый коллажный шаблон.jpg* (меню **Файл** | **Открыть**);

• при необходимости увеличьте на шаблоне область лица персонажа при помощи инструмента “лупа” ;

• выберите инструмент “Волшебная палочка” и щелкните на закрашенной одним цветом области лица персонажа (той, куда вы собираетесь подставить свое лицо). С помощью инструмента “Точность” Точность: 50% , щелкая мышью в соответствующем месте полоски, установите точность выделения областей “волшебной палочкой” и снова попробуйте выделить нужную область. В результате на шаблоне должна быть выделена именно та часть лица персонажа, на месте которой должно быть ваше лицо;

• нажмите клавишу **Delete** (или **Del**) на клавиатуре. Выбранная область станет прозрачной — в графическом редакторе она выглядит как шахматное поле из белых и серых квадратиков. Повторите выделение и стирание всех таких областей;

Рис. 6

• откройте фотографию *Лицо.jpg* (меню **Файл | Открыть**). Если вы будете использовать собственную фотографию, то она должна быть такой, чтобы расположение лица на ней примерно соответствовало расположению лица персонажа на шаблоне (например, если персонаж смотрит прямо, а его голова чуть наклонена, то лицо на фотографии тоже должно смотреть прямо и тоже должно быть наклонено);

• выделите нужную часть своей фотографии (лицо и некоторую область вокруг него) инструментом выделения и скопируйте этот фрагмент в буфер обмена (меню **Правка | Копировать**);

• перейдите к шаблону фоторамки и выполните команду меню **Правка | Вставить в новый слой**. При необходимости выберите в окне запроса пункт `Expand canvas`;

• используя кнопку в окне **Слои**, переместите слой с вставленной фотографией вниз, под слой шаблона;

• проверьте, чтобы в окне **Слои** был выделен слой с фотографией (для выделения слоя достаточно щелкнуть на его имени мышью);

• подгоните фрагмент фотографии с вашим лицом по размерам и расположению так, чтобы она просвечивала сквозь прозрачную область шаблона и выглядела наиболее естественно;

• закончив работу, сохраните получившуюся коллажную фотографию с помощью команд **Файл | Сохранить как**. Задайте имя файла *Коллажная фотография*, а расширение — *png*. Нажмите кнопку **Сохранить**;

• появится диалоговое окно **Слить слои?**. Выберите в нем `Flatten` (“Объединение”);

• если при вставке фотографии вы выбирали `Expand canvas`, выделите полученную картинку с помощью инструмента прямоугольного выделения и обрежьте белые поля с помощью инструмента “Обрезать по выделенному”;

• сохраните результат (меню **Файл | Сохранить**).

Коллажная фотография готова.

Рис. 7

3. Фотомонтаж: подготовка коллажного шаблона и вставка в него фотографии (см. *рис*. 9)

Чтобы поместить фрагмент фотографии (например, ваше лицо) в коллажный шаблон, подготовленный вами из произвольной фотографии (см. *рис*. 8):

• откройте файл *Шаблон.jpg*;

• увеличьте на этой фотографии область лица заснятого человека/персонажа при помощи инструмента “Лупа”;

• выберите инструмент “Ластик” и удалите с помощью этого инструмента часть лица для создания собственного шаблона;

• очищенная область будет прозрачной — в графическом редакторе она выглядит как шахматное поле из белых и серых квадратиков;

• сохраните полученный шаблон: меню **Файл | Сохранить как**, имя файла *Коллажный шаблон*, расширение файла — *pdn* (стандартное для программы *Paint.NET*). Нажмите кнопку **Сохранить**;

Рис. 8

• откройте фотографию *Конкурс.jpg* (меню **Файл | Открыть**). Фотография должна быть такой, чтобы расположение лица на ней примерно соответствовало расположению лица персонажа на шаблоне (например, если персонаж смотрит прямо, а его голова чуть наклонена, то ваше лицо на фотографии тоже должно смотреть прямо и тоже должно быть наклонено);

• выделите нужную часть своей фотографии (лицо и некоторую область вокруг него) инструментом выделения и скопируйте этот фрагмент в буфер обмена (меню **Правка | Копировать**);

• перейдите к шаблону и выполните команду меню **Правка | Вставить в новый слой**. При необходимости выберите в окне запроса пункт `Expand canvas`;

• используя кнопку в окне **Слои**, переместите слой с вставленной фотографией вниз, под слой шаблона;

• проверьте, чтобы в окне **Слои** был выделен слой с фотографией (для выделения слоя достаточно щелкнуть на его имени мышью);

• подгоните фрагмент фотографии с вашим лицом по размерам и расположению так, чтобы она просвечивала сквозь прозрачную область шаблона и выглядела наиболее естественно;

• закончив работу, сохраните получившуюся коллажную фотографию с помощью команд **Файл** | **Сохранить как**. Задайте имя файла *Коллажная фотография*, а расширение — *png*. Нажмите кнопку **Сохранить**;

• появится диалоговое окно **Слить слои?**. Выберите в нем `Flatten` (“объединение”);

• если при вставке фотографии вы выбирали `Expand canvas`, выделите полученную картинку с помощью инструмента прямоугольного выделения и обрежьте белые поля с помощью инструмента “Обрезать по выделенному” . Сохраните результат.

Коллажная фотография готова.

Рис. 9

4. Фотомонтаж: создание художественной фоторамки (см. *рис*. 13)

Запустите программу Paint.Net.

Откройте фон из файла *5.jpg*. Убедитесь, что фоновая картинка появилась в окне **Слои**:

Рис. 10

Наложите поверх фона фоторамку:

• откройте файл рамки *1.jpg*;

• выберите инструмент “Волшебная палочка” и щелкните на закрашенной белым цветом области фона вокруг фоторамки. С помощью инструмента Точность: 50% установите точность выделения внешней области “волшебной палочкой” и снова попробуйте выделить эту область. В результате у вас должен быть выделен только фон вне рамки, но не сама рамка и не область внутри нее;

• нажмите клавишу **Delete** (**Del**) на клавиатуре — выбранная область станет прозрачной (в графическом редакторе она выглядит как шахматное поле из белых и серых квадратиков);

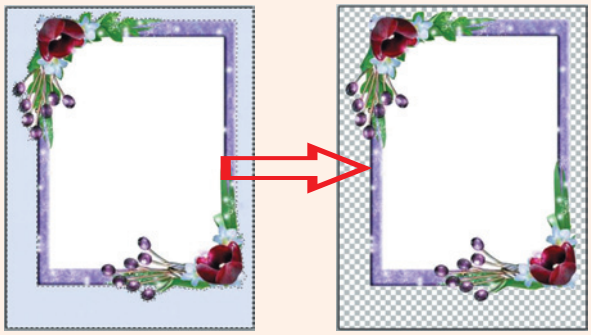


Рис. 11

• выделите фоторамку инструментом выделения и скопируйте ее в буфер обмена (меню **Правка** | **Копировать**);

• перейдите к фону и выполните команду меню **Правка** | **Вставить в новый слой**. При необходимости выберите в окне запроса пункт `Expand canvas`. Вставленная рамка должна располагаться в окне **Слои** над фоном, и этот слой с рамкой должен быть выделен:

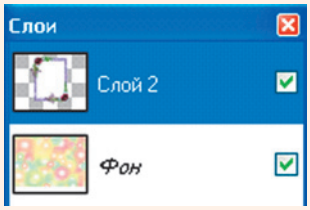


Рис. 12

• измените размеры и расположение рамки по своему желанию.

Добавьте оформление в виде сказочных персонажей:

• откройте файл со слоненком (*3.jpg*);

• при помощи инструмента “Волшебная палочка” (при необходимости настраивая его точность) и клавиши **Delete** сделайте прозрачным фон вокруг персонажа — аналогично тому, как вы это только что делали для рамки. Если фон вокруг персонажа представляет собой несколько отдельных замкнутых областей, то сделайте прозрачной каждую такую область;

• выделите рисунок персонажа инструментом выделения и скопируйте в буфер обмена (меню **Правка** | **Копировать**);

• перейдите к фону и выполните команду меню **Правка** | **Вставить в новый слой**. При необходимости выберите в окне запроса пункт **Expand canvas**. Вставленный рисунок в окне **Слои** образует еще один слой (он будет выделен);

• меняя размеры и положение персонажа, разместите его рядом с рамкой (в том числе так, что он будет ее частично закрывать);

• откройте файл с изображением кенгуренка (файл *17.png*). Здесь прозрачный фон вокруг рисунка уже есть;

• выберите в меню **Рисунок** пункт **Flip Horizontal** (“Отразить по горизонтали”);

• выделите рисунок, скопируйте его в буфер обмена и вставьте на фон как новый слой — аналогично предыдущему персонажу. Смасштабируйте и разместите его в нужном месте. Обратите внимание: этот персонаж очень велик, поэтому при его вставке картинка с фоном сильно уменьшится. Сначала уменьшите кенгуренка до размеров картинки с фоном, а затем используйте инструмент "Лупа" , чтобы увеличить рисунок;

• повторите описанные операции для рисунка "морской звезды" (файл *3.png*). Разместите ее в верхней части создаваемой вами художественной фоторамки;

• если персонажи взаимно накладываются друг поверх друга, то меняйте их взаимное перекрытие, используя кнопки в окне **Слои**: слой с изображением, которое должно перекрывать другое изображение, в окне **Слои** должен быть выше перекрываемого слоя.

Закончив работу, сохраните получившуюся фоторамку с помощью команды **Файл | Сохранить как**. Задайте имя файла *Моя фоторамка*, а расширение — *pdn*. Этот формат файла сохраняет разделение на слои, так что позже вы сможете вновь открыть этот файл и изменить расположение рамки и персонажей, заменить какие-то из них или общий фон и т.д.

Сохраните фоторамку еще раз командой **Файл | Сохранить как**, но выберите расширение файла *png*. Нажмите кнопку **Сохранить**. В диалоговом окне **Слить слои?** выберите **Flatten** ("Объединение"). Выделите полученную картинку с помощью **инструмента прямоугольного выделения** и обрежьте белые поля с помощью инструмента "Обрезать по выделенному" . Сохраните результат.

Художественная фоторамка готова (*рис.* 13). Вставьте в нее фотографию (*файл 4.jpg*), как описано в разделе 1 (*рис.* 14).

Рис. 13

Рис. 14

5. Фотомонтаж: выделение объектов сложной формы, создание фотоколлажа (см. *рис.* 18)

Запустите программу Paint.Net.

Откройте фон из файла *рыцарь1.jpg*.

Ваша цель — выделить фигуру рыцаря, отделив ее от остального изображения (фона и других изображений):

• для выделения объекта по сложному контуру используется инструмент "Лассо" ;

• режимы выделения областей инструментом "Лассо", а также другими инструментами выделения — прямоугольной области и овальной области:

— для выделения области нажмите левую кнопку мыши и начните двигать мышь — по мере ее движения будет формироваться замкнутая серая область выделения (ранее выделенные области при этом сбрасываются); после отпускания мыши область выделения отображается "бегущим" пунктирным контуром:

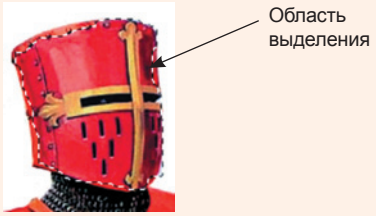


Рис. 15

— если нажать и удерживать клавишу Ctrl, а затем начать выделять область при помощи левой кнопки мыши, то новая область выделения добавляется к ранее созданной:

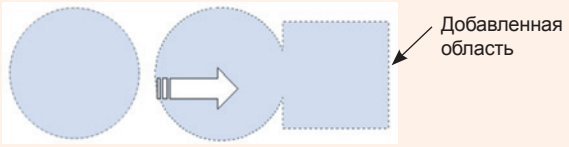


Рис. 16

— если выделять область при помощи правой кнопки мыши, то это позволяет снимать выделение в пределах рисуемой "Лассо" области:

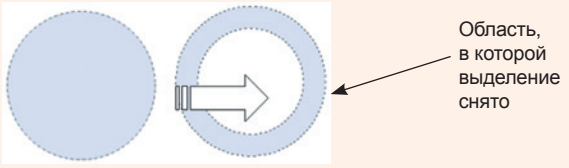


Рис. 17

— чтобы повысить точность выделения, выберите инструмент "Лупа" и щелкните левой кнопкой мыши для увеличения масштаба (щелчок правой кнопкой мыши уменьшает масштаб); то же позволяет делать колесико мыши при нажатой клавише Ctrl; для продолжения работы с выделением снова выберите инструмент "Лассо" либо другой инструмент выделения;

— инструмент "Серая стрелка" позволяет перемещать мышью изображение, выделенное областью выделения (причем в окне со списком слоев

должен быть выбран именно нужный слой!); инструмент “Белая стрелка” перемещает только саму область выделения, позволяя более точно “нацеливать” ее (накладывать на изображение) и масштабировать область выделения;

• установив более крупный масштаб изображения ( ), выделите рыцаря, используя операции добавления и снятия выделения (клавиша Ctrl и правая кнопка мыши), а также перемещая рисунок в окне Paint.Net с помощью линеек прокрутки или (вверх/вниз) колесиком мыши;

> ***Внимание!*** *Если вы щелкнете на рисунке левой кнопкой мыши, не удерживая нажатой клавишу Ctrl, то все сделанное выделение сбросится, и надо будет начать все с начала.*

• завершив выделение, скопируйте выделенную часть рисунка в буфер обмена (меню **Правка | Копировать**). Вставьте этот скопированный фрагмент как новый рисунок (меню **Правка | Вставить в новое изображение**). Фрагмент вставится на прозрачном фоне. Просмотрите его и ластиком “подчистите” его, стирая ненужные остатки изображения;

• сохраните готовый рисунок в файл в формате Paint.Net;

• аналогичным способом откройте файл *Honda_OSM_2008_32.jpg*. Выделите автомобиль, скопируйте его, вставьте в новый слой и сохраните в файле формата *pdn*;

• откройте файл *марс1.jpg*;

• скопируйте на этот рисунок ранее созданные фрагменты с рыцарем и автомашиной, вставляя их как новые слои. Управляя порядком следования слоев, перемещая и масштабируя их, создайте картинку-коллаж: “Средневековый рыцарь приехал отдохнуть на Марс в своем автомобиле” ☺.

Рис. 18

ЯПОНСКИЙ УГОЛОК

Две судоку

Решите, пожалуйста, две японские головоломки “судоку”:

1) простую:

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

2) сложную:

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

Ответы присылайте в редакцию (можно решать не все судоку).

ПОИСК ИНФОРМАЦИИ

Четыре вопроса

Ответьте, пожалуйста, на четыре вопроса.

1. Кто из классиков русской поэзии заседал в Чрезвычайной комиссии по расследованию преступлений царского режима?
2. Какое растение идет на создание домов, мостов, мебели, зонтиков и вееров?
3. Как назывался военный плащ древних римлян?
4. Как звали морскую принцессу из диснеевского мультфильма “Русалочка”?

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

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

Джордж Пойя

Компьютер помогает искать преступника. Часть 2

В предыдущем выпуске была описана методика разработки программы, с помощью которой можно решить такую задачу: “Пять свидетелей оказались на месте преступления, хорошо разглядели бежавшего преступника и согласились помочь криминалистам составить его словесный портрет. Каждый из свидетелей дал свое описание. Вот они:

1) черные волосы, маленький нос, борода, маленькие глаза, круглое лицо;

2) рыжие волосы, большой нос, борода, маленькие глаза, круглое лицо;

3) черные волосы, большой нос, чисто выбрит, маленькие глаза, круглое лицо;

4) лысый, большой нос, чисто выбрит, большие глаза, худое лицо;

5) рыжие волосы, маленький нос, борода, большие глаза, круглое лицо.

Каждый из свидетелей правильно указал по меньшей мере одну примету преступника и допустил одинаковое с остальными число ошибок.

Как составить точный словесный портрет преступника?”.

Теперь решим задачу с помощью программы Microsoft Excel или подобной.

Закодируем все возможные значения отдельных признаков следующим образом:

1) волосы:

— черные волосы — 1;

— рыжие волосы — 2;

— лысый — 3;

2) нос:

— маленький нос — 1;

— большой нос — 2;

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		**Словесный портрет**											
2													
3		Вариант свидетеля											
4	1-го	1	1	1	1	1							
5	2-го	2	2	1	1	1							
6	3-го	1	2	2	1	1							
7	4-го	3	2	2	2	2							
8	5-го	2	1	1	2	1							
9													
10							**Степень совпадения**						
11		Вариант сочетания примет					*1-го свидетеля*					Кпр	Кош
12	*1*	1	1	1	1	1	1	1	1	1	1	5	0
13	*2*	1	1	1	1	2	1	1	1	1	0	4	1
14	*3*	1	1	1	2	1	1	1	1	0	1	4	1
15	*4*	1	1	1	2	2							
16	*5*	1	1	2	1	1							
…													
26	*15*	1	2	2	2	1							
27	*16*	1	2	2	2	2							
28	*17*	2	1	1	1	1							
29	*18*	2	1	1	1	2							
30	*19*	2	1	1	2	1							
…													
42	*31*	2	2	2	2	1							
43	*32*	2	2	2	2	2							
44	*33*	3	1	1	1	1							
45	*34*	3	1	1	1	2							
46	*35*	3	1	1	2	1							
…													
58	*47*	3	2	2	2	1							
59	*48*	3	2	2	2	2							
60													

3) борода:
— есть борода — 1;
— чисто выбрит — 2;
4) глаза:
— маленькие глаза — 1;
— большие глаза — 2;
5) лицо:
— круглое лицо — 1;
— худое лицо — 2.

В этой “системе кодировки” показания каждого из свидетелей будут выглядеть так:

1) первый свидетель: 11111;
2) второй свидетель: 22111;
3) третий свидетель: 12211;
4) четвертый свидетель: 32222;
5) пятый свидетель: 21121.

Оформим лист так, как показано на рисунке ниже.

В диапазоне ячеек B4:F8 запишем только что приведенные показания каждого из свидетелей (пять элементов портрета), а в диапазоне B12:F59 — все возможные варианты сочетания примет. Несмотря на то что всего вариантов — 48, эта работа не является трудоемкой, поскольку большая часть значений может быть получена путем копирования. Так, в столбце F повторяются значения 1 и 2, в столбце E — блок из четырех значений, в столбце D — из восьми значений, в столбце C — из 16 значений.

Далее, как описано в предыдущем выпуске, для каждого из 48 возможных вариантов портрета нужно определить число ошибок у каждого свидетеля. Эта работа не сложная по содержанию, но несколько трудоемкая.

Итак, начнем. Для первого свидетеля:

1) в ячейку G12 запишем формулу, определяющую степень совпадения первого признака портрета (волосы) у этого свидетеля:

=ЕСЛИ(B12=B$4;1;0)

Видно, что степень совпадения задается значениями 1 и 0;

2) вид приведенной формулы (в ней использованы относительная и смешанная ссылки) позволяет скопировать ее на другие признаки и на другие варианты сочетания примет — в диапазон ячеек G12:K59;

3) в ячейки L12 и M12 запишем формулы, подсчитывающие, соответственно, число правильных элементов и число ошибок для первого варианта словесного портрета:

=СУММ(G12:K12)

и

=5 – L12;

4) скопируем эти две формулы в диапазон ячеек L13:M59.

Аналогично заполним ячейки:

— для второго свидетеля (N12:T59); формула в ячейке N12:

=ЕСЛИ(B12=B$5;1;0)

копируется в диапазон ячеек N12:R59;

— для третьего свидетеля (U12:AA59); формула в ячейке U12:

=ЕСЛИ(B12=B$6;1;0)

копируется в диапазон ячеек U12:Y59;

— для четвертого свидетеля (AB12:AH59); формула в ячейке AB12:

=ЕСЛИ(B12=B$7;1;0)

копируется в диапазон ячеек AB12:AF59;

— для пятого свидетеля (AI12:AO59); формула в ячейке AI12:

=ЕСЛИ(B12=B$7;1;0)

копируется в диапазон ячеек AI12:AM59.

После этого можно найти вариант словесного портрета, соответствующий фактическому портрету преступника. Это вариант, удовлетворяющий условию — количество ошибок у каждого свидетеля одинаково и при этом не равно 5. Для его нахождения следует сравнить значения в столбцах M, T, AA, AH и AO, используя функции ЕСЛИ и И.

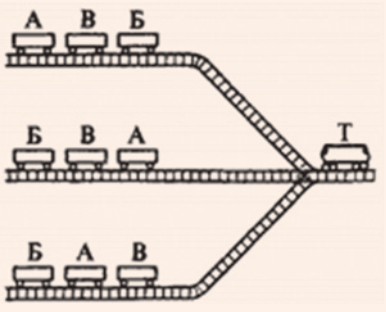
Задание для самостоятельной работы

Оформив лист, как описано, найдите правильный вариант словесного портрета. Лист и словесное описание портрета присылайте в редакцию. Все приславшие правильный и обоснованный ответ будут награждены дипломами. Срок представления ответов — октябрь.

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

Головоломка “Переставить вагоны”

Напомним, что нужно было разработать алгоритм действий машиниста тепловоза Т (см. рисунок ниже) для перестановки вагонов с арбузами (А), бананами (Б) и виноградом (В) таким образом, чтобы на каждом из путей оказались составы с одинаковыми плодами.



Задача может быть решена несколькими способами. Приведем два из них:

1)

Исх.	1	2	3	4	5	6	7	8
АВБ	А	АА	АА	ААА	ААА	АААВВВ	АААВВВ	ААА
БВА	БВА	БВ	БВВВБ	БВВВБ	БВВВ	Б		ВВВ
БАВ	БАВВБ	БАВВБ	БА	Б	ББ	ББ	ББВ	ББВ

2)

Исх.	1	2	3	4	5	6	7	8
АВБ	АВБ	АВ	АВ	АВ	АВВ	А	ААА	ААА
ББА	ББААВ	ББААВ		БББВААВ	БББВАА	БББВАА	БББВ	ББВ
БАВ	Б	ББ	БББВААВ			ВВ	ВВ	ВВВ

Ответы представили:

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

— Алехина Елена, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Базанов Илья и Миноцкий Ян, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.** (Илья и Ян разработали два варианта решения);

— Бородюк Анна и Василенко Татьяна, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

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

— Добрякова Светлана, Козина Мария и Чернова Наталья, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

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

— Иванова Виолетта и Левченко Ирина, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

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

Софизм “Все числа равны между собой”

Напомним, что предлагалось найти ошибку в следующем софизме (рассуждении, “внешне” правильном, но противоречащем общепринятым представлениям).

Пусть $m \neq n$. Возьмем тождество:

$$m^2 - 2mn + n^2 = n^2 - 2nm + m^2.$$

Имеем $(m - n)^2 = (n - m)^2$. Отсюда $m - n = n - m$, или $2n - 2m$, а значит, $n = m$ (!?).

Объяснение

Если квадраты чисел равны, то это еще не означает, что и сами числа равны. Из равенства квадратов двух чисел вытекает лишь, что равны абсолютные величины этих чисел. Например, если $4 = 4$, то извлечение квадратного корня из обеих частей означает, что $2 = 2$ либо $-2 = -2$, но не означает, что $2 = -2$. Поэтому ошибка в рассуждении, начинающемся словом “Отсюда…”.

Правильные ответы прислали:

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

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

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

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

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

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

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

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

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

— Сорокина Дарья, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

Головоломка “Числовой ребус с ЗАЗОРОМ”

Напомним, что требовалось решить числовой ребус:

$$\begin{array}{cccccc} & & & & & \mathbf{Р} \\ & & & & \mathbf{О} & \mathbf{Р} \\ + & & & \mathbf{З} & \mathbf{О} & \mathbf{Р} \\ & & \mathbf{А} & \mathbf{З} & \mathbf{О} & \mathbf{Р} \\ & \mathbf{З} & \mathbf{А} & \mathbf{З} & \mathbf{О} & \mathbf{Р} \\ \hline & 5 & 5 & 5 & 5 & 0 \end{array}$$

Решение

Анализ крайнего левого разряда показывает, что **З** = 5 (при этом **А** = 2) или **З** = 4 (**А** = 7).

Исследуем разряд сотен. Здесь **З** + **З** + **З** + (“в уме” из разряда десятков). Допустим, **З** = 5. Тогда переноса из разряда десятков нет, а это возможно при **О** = 1 и при **Р** = 2. Но последнее равенство невозможно, так как **А** = 2. Значит, **З** = 4. Запишем известные цифры в ребус:

$$\begin{array}{cccccc} & & & & & \mathbf{Р} \\ & & & & \mathbf{О} & \mathbf{Р} \\ + & & & 4 & \mathbf{О} & \mathbf{Р} \\ & & 7 & 4 & \mathbf{О} & \mathbf{Р} \\ & 4 & 7 & 4 & \mathbf{О} & \mathbf{Р} \\ \hline & 5 & 5 & 5 & 5 & 0 \end{array}$$

Далее. Видно, что **О** = 8 (только в этом случае сумма в разряде десятков равна 35, и в разряд сотен переносится 3). Все решение ребуса:

```
          6
        8 6
+     4 8 6
    7 4 8 6
4 7 4 8 6
-----------
5 5 5 5 0
```

Правильный ответ прислали:

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

— Афанасьева Ксения, Демьянова Елена и Сорокина Анна, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

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

— Бляхина Ольга, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

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

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

— Жукова Людмила, Нестерова Анастасия, Одинцова Екатерина, Попова Екатерина и Хажиев Флорит, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Казакова Ирина, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Козина Мария, Локтев Иван и Чернова Наталья, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

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

— Сорокина Дарья, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**

Задача “Блондины и брюнеты в классе”

Напомним условие: “В классе девочек-блондинок столько же, сколько мальчиков-брюнетов. Кого в классе больше — девочек или учащихся с темными волосами? Принять, что шатенов и шатенок, а также учащихся с другим цветом волос в классе нет ☺”.

Решение

Очевидно, что класс состоит из:

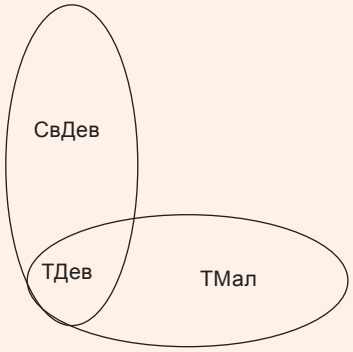
1) девочек-блондинок;

2) мальчиков-блондинов;

3) девочек-брюнеток;

4) мальчиков-брюнетов.

Общее количество девочек в классе есть сумма числа девочек-блондинок и числа девочек-брюнеток. Общее число брюнетов есть сумма количества мальчиков-брюнетов и числа девочек-брюнеток. Но первые слагаемые этих сумм равны по условию, а вторые слагаемые совпадают (см. рисунок ниже, на котором брюнеты обозначены “Т”, а блондины — “Св”).



Итак, ответ — девочек в классе столько же, сколько всего учащихся с темными волосами.

Правильные ответы прислали:

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

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

— Воротникова Елизавета и Катышева Людмила, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

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

— Закатов Владислав, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

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

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

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

— Сорокина Дарья, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**

Задача “Сын профессора Алгоритмова”

Напомним, что предлагалось определить точную дату рождения сына профессора Бита Байтовича Алгоритмова, если известно, что:

1) день рождения сына является числом Фибоначчи;

2) сумма цифр числа — дня рождения и произведение его цифр также являются числами Фибоначчи;

3) хотя порядковый номер месяца, в котором родился сын, и не является числом Фибоначчи, зато является произведением двух не соседних чисел Фибоначчи;

4) год рождения сына представляет собой удвоенное число Фибоначчи.

Решение

Числами Фибоначчи называют числа, образующие последовательность 1, 2, 3, 5, 8, … (каждый член последовательности, начиная с третьего, равен сумме двух предыдущих).

Из условия следует, что день рождения сына профессора — двузначное число. Так как номер дня месяца не может быть больше 31, то днем рождения может быть 13-е или 21-е число. Первое значение не подходит, так как сумма его цифр также является числом Фибоначчи. Значит, сын родился 21-го числа.

Номер месяца, в котором родился сын профессора, не может быть больше 12. Из первых пяти чисел Фибоначчи (1, 2, 3, 5, 8) условию соответствуют "несоседние" числа 2 и 5, то есть мальчик родился в октябре.

Год рождения можно определить, используя, например, электронную таблицу Microsoft Excel или подобную.

	A	B
1	Числа Фибоначчи	Удвоенные значения
2	1	2
3	2	4
…		
15	610	1220
16	987	1974
17	1597	3194

Ответ: искомая дата рождения — 21.10.1974.

Ответы прислали:

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

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

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

— Баранец Дарья, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

— Ботова Дарья, Танасюк Артем, Калугин Данил, Миронова Екатерина, Молева Александра, Чирочкина Мария и Чурасов Михаил, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Воротникова Елизавета и Катышева Людмила, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Дубова Валерия, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Жарова Светлана, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Закатов Владислав, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

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

— Коршаков Вадим и Олешева Дарья, Смоленская обл., г. Демидов, школа № 1, учитель **Кордина Н.Е.**;

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

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

Задание "Две шарады и логогриф"

Ответы

Шарада № 1: автомат (а + в + томат).

Шарада № 2: компьютер (ком + пью + тер).

Логогриф: экран (э + кран).

Ответы прислали:

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

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

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

— Базанов Илья и Кротов Олег, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Битюгова Елена и Чернова Мария, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

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

— Герасимова Наталья и Костина Евгения, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

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

— Иванова Ксения и Мухина Светлана, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Коростелев Иннокентий и Марун Виталий, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Краснова Елена, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

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

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

— Рюмина Екатерина, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Якуба Наталья, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**

Задача “Расположить ребят по росту”

Напомним, что требовалось расположить имена четырех ребят — Андрея, Бориса, Ивана и Григория в порядке возрастания их роста, если известно, что Борис не самый высокий, но он выше Андрея и Григория, а Андрей ниже Григория.

Ответ: Андрей, Григорий, Борис и Иван.

Правильные ответы прислали:

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

— Битюгова Елена и Чернова Мария, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

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

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

— Иванова Ксения и Мухина Светлана, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Коростелев Иннокентий и Марун Виталий, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Рюмина Екатерина, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

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

— Федорченко Александра, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Цыганков Евгений, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Цыганкова И.Ю.** (Евгений решил задачу методом схем);

— Якуба Наталья, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**

Задача “На катке”

Напомним условие: “Четыре подруги пришли на каток каждая со своим братом. Они разбились на пары и начали кататься. Оказалось, что в каждой паре “кавалер” выше “дамы” и никто не катался со своей сестрой. Самый высокий из компании — Юра Воробьев, следующий по росту — Андрей Егоров, потом Люся Егорова, Сережа Петров, Оля Петрова, Дима Крымов, Инна Крымова и Аня Воробьева. Кто с кем катался?”

Ответ: Юра Воробьев катался с Люсей Егоровой, Андрей Егоров — с Олей Петровой, Сережа Петров — с Инной Крымовой, Дима Крымов — с Аней Воробьевой.

Правильные ответы представили:

— Абрамова Светлана, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

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

— Базанов Илья, Головченко Тихон, Кротов Олег, Манукян Григорий и Миноцкий Ян, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Бакланова Елена, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

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

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

— Глушаков Андрей, Семак Татьяна и Царева Татьяна, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Закатов Владислав, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

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

— Коршаков Вадим и Олешева Дарья, Смоленская обл., г. Демидов, школа № 1, учитель **Кордина Н.Е.**;

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

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

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

— Мучник Анна и Прихожан Ирина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Усанина Светлана, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**

Головоломка “Крест-накрест”

Решение

Р	А	С	Т	Р
П	О	Т	О	К
Р	Е	Б	Р	О
В	Ы	В	О	Д
В	О	Л	Ь	Т

Правильные ответы прислали:

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

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

— Бакланова Елена, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

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

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

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

— Дерюгина Елизавета и Катышева Людмила, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Дружинина Анастасия, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Красненкова Л.А.**;

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

— Климук Анастасия, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

— Лёвина Татьяна, Макаренко Светлана и Томилин Александр, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Лопаткина Мария и Храмчук Елизавета, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

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

— Тупицына Лидия, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**

Новогодний кроссворд

Ответы

По горизонтали: 2. Фон. 3. Слово. 4. Дискета. 8. Атом. 9. Зеро. 10. Единица. 13. Экран. 14. Цифра. 17. Адресация. 19. Игра. 20. Абак. 21. Буфер. 22. Хакер. 23. Поток. 25. Вывод. 26. Пилотка. 29. Адаптер. 30. Запятая. 31. Диск. 32. Цикл. 33. Альфа.

По вертикали: 1. Дорожка. 4. Домен. 5. Абзац. 6. Старт. 7. Шрифт. 11. Интеграл. 12. Иерархия. 15. Логотип. 16. Драйвер. 17. Амулет. 18. Ячейка. 24. Килобит. 25. Вставка. 27. Ада. 28. Связь. 29. Ара.

Правильные ответы представили:

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

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

— Ахматгалиева Диана, Вольф Николай, Одинцова Екатерина, Пуговкина Ксения, Танасюк Артем, Чаусов Евгений и Шелуханова Татьяна, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

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

— Бакланова Елена и Землякова Ирина, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Букин Сергей и Цаплина Ирина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Воротникова Елизавета и Катышева Людмила, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

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

— Лёвина Татьяна, Макаренко Светлана и Томилин Александр, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Новиков Владимир и Пастухов Олег, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

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

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

— Шаповаленко Анастасия, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

Решение задач из старинного учебника математики, который был написан в VII веке армянским ученым Анания Ширакаци, прислали:

— Ботова Дарья, Галиуллин Антон, Коптева Алена, Меньшенина Ксения и Танасюк Артем, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

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

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

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

Правильное решение задачи, связанной с передачей файла по каналам связи за наименьшее время от одного мальчика к другому, представили:

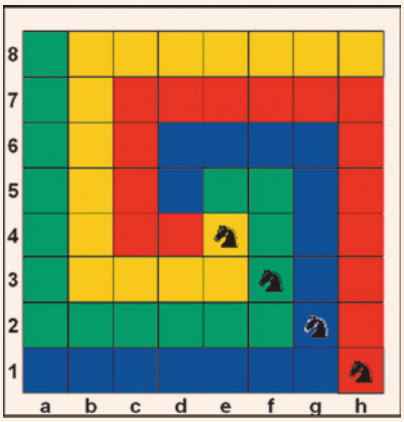
— Анюков Дмитрий, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

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

— Галиуллин Антон, Калугин Данил и Орлов Кирилл, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**

Оптимальный маршрут передачи: Павел — Ахмед — Семен — Бадри — Владимир (длительность передачи — 38 секунд).

Правильное решение головоломки “Расстановка четырех шахматных фигур” представили Ахматгалиева Диана и Шелуханова Татьяна, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.** (решение показано на рисунке ниже).



Правильное решение задачи “Кто на ком женат?” представили также Грибанов Владлен, Дукач Светлана, Лысенко Екатерина, Овчинникова Елизавета и Соболев Иван, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**

Задания, предложенные для самостоятельной работы в статье “Восточный календарь — своими руками”, выполнила Кузнецова Юлия, г. Ярославль, школа № 33, учитель **Ярцева О.В.** Она разработала два варианта компьютерной программы, с помощью которой можно определить, какому году по восточному календарю соответствует тот или иной год нашего летосчисления. С их помощью Юлия установила, что 1949 год был годом Желтой Коровы, а 2013 год является годом Черной Змеи. Юлия будет награждена дипломом.

Ребус из звездочек, опубликованный в декабрьском выпуске, правильно решила Бурмантова Юлия, ученица 6-го класса Совхозной средней школы (Московская обл., Серебряно-Прудский р-н, поселок Успенский, учитель **Жарикова Е.Н.**). Она также будет награждена дипломом.

Поздравляем!

Победителями конкурса № 97 “Информатика на шахматной доске” стали:

— Бойкова Юлия, г. Воронеж, лицей № 2, учитель **Комбарова С.И.**;

— Василенко Ирина, ученица 3-го класса средней школы села Горелово Тамбовской обл., учитель **Дудкина Л.А.**;

— Воропаев Андрей, г. Воронеж, лицей № 2, учитель **Гаврилова И.В.**;

— Гавриков Александр, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Медведьков Роман, ученик 1-го (!) класса гимназии № 40 г. Калининграда, учитель **Медведькова Н.А.**;

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

— Семенова Елизавета, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**

Все они будут награждены дипломами.

В мартовском выпуске “В мир информатики” были подведены итоги конкурса № 96 (его задание было связано с решением задачи перемещения мальчиков на лифте). Одним из победителей этого конкурса стала Власенко Мария, студентка Ардатовского коммерческо-технического техникума, поселок Ардатов Нижегородской обл., преподаватель **Зудин В.П.**

Дипломами будут награждены также Гладких Владислава и Пронина Вероника, ученицы лицея № 4 им. Героя Советского Союза Г.Б. Злотина, г. Орел (учитель **Чапкевич И.М.**), представившие результаты работы над проектами “Палиндромы” и “Числа Фибоначчи”.

Поздравляем всех награжденных!

КРЕПКИЙ ОРЕШЕК

Головоломка “Россия — чемпион! ☺”

Напомним, что в головоломке требовалось восстановить таблицу результатов встреч четырех команд (указать, с каким счетом сыграли команды между собой) по известной итоговой таблице:

	В	Н	П	Мячи	Очки
Россия	2	1	0	9–1	5
Испания	2	1	0	5–1	5
Италия	1	0	2	2–9	2
Португалия	0	0	3	0–5	2

Каждая команда провела с каждой по одному матчу.

Благодаря следующих читателей, приславших правильный ответ:

— Грибанова Владлена, Дукач Светлану, Лысенко Екатерину, Овчинникову Елизавету и Соболева Ивана, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

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

— Танасюка Артема, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**,

приведем таблицу результатов без указания счета каждого матча:

					В	Н	П	Мячи	Очки
Россия		Н	В	В	2	1	0	9–1	5
Испания	Н		В	В	2	1	0	5–1	5
Италия	П	П		В	1	0	2	2–9	2
Португалия	П	П	П		0	0	3	0–5	0

Примечание. “В” — выигрыш, “Н” — ничья, “П” — поражение.

Предлагаем восстановить таблицу с указанием конкретных результатов игр. Ответы присылайте в редакцию.

Восемь семейных пар (задачка “на лето”)

Восемь семейных пар встретились, чтобы обменяться книгами. Мужья и жены, как это принято, носили одну и ту же фамилию; к тому же они имели одинаковые профессии и водили одинаковые марки автомобиля. У каждой пары был свой любимый цвет. Кроме того, известно, что:

1. Маша Иванова и ее муж работают продавцами-консультантами.
2. Книгу “Турецкий гамбит” принесла пара, у которой “Хонды” и оба они любят красный цвет.
3. Сережа и Катя, его жена, любят коричневый цвет.
4. Люба и Юра Егоровы обожают белый цвет.
5. Света Петрова и ее муж работают логистиками и водят “Шкоды”.
6. Наташа и ее муж Игорь взяли почитать “Ярмарку тщеславия”.
7. Саша и его жена любят розовый цвет и принесли “Пеппи Длинныйчулок”.
8. Ира и Данила — менеджеры.
9. “Алису в Зазеркалье” позаимствовала пара, которая водит “Форды”.
10. Зайцевы — юристы. Они принесли роман “Парфюмер”.
11. Сидоровы — врачи. Они купили “Фиесты”.
12. Иван и его жена любят зеленый цвет.
13. Галине Соловьевой и ее мужу нравится синий цвет.
14. Константин и его жена принесли “Фиесту”. Они водят “Лексусы”.
15. Одна пара принесла книгу “Три мушкетера”, а взяли почитать “Пеппи Длинныйчулок”.
16. Пара, которая водит “Субару”, любит фиолетовый цвет.
17. Бухгалтеры взяли почитать “Три мушкетера”.
18. Кассиры водят “Ниссан”.
19. Анна и ее муж водят “Опель” и принесли “Ярмарку тщеславия”.
20. Они же взяли почитать книгу, которую принесли Хабаровы.
21. Коля и его жена любят желтый. Они взяли почитать “Сто лет одиночества”.
22. Кривцовы работают программистами.
23. Книга “Сто лет одиночества” была принесена парой, которая водит “Лады”.

Определите, как зовут супругов в каждой паре, каковы их фамилии и марки машин, какой цвет они любят, какую книгу принесли, а какую взяли почитать, кем они работают.

Срок представления ответов — 1 октября. Все приславшие правильный ответ с его обоснованием будут награждены дипломами.

Уважаемые коллеги!

Для поощрения самых активных участников конкурсов, проводимых в разделе “В мир информатики”, редакция может направить вам электронный вариант диплома.

Заявку на диплом просьба прислать в редакцию электронной (адрес: *vmi@1september.ru*) или обычной почтой в апреле–июне. Оформление дипломов будет проводиться в учебном заведении.

Об эмблеме рубрики “Японский уголок”

В редакцию пришло письмо от учителя математики и информатики средней школы села Восточное Нижегородской области Галины Александровны Долговой. Она пишет о том, что ее ученики спрашивают, как переводится на русский язык иероглиф — эмблема рубрики “Японский уголок”.

Отвечаем — данный иероглиф означает *друг* (выбирая его, редакция имела в виду каждого из вас, дорогие читатели ☺).

Уже после подготовки списков читателей, приславших ответы на задания наших конкурсов, были получены письма от:

— Киришиной Татьяны, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**, которая правильно решила японские головоломки “судоку”, опубликованные в декабрьском выпуске;

— Трептау Татьяны, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Цыганкова И.Ю.**; Татьяна правильно решила головоломку “Числовой ребус с ЗАЗОРОМ”, а также одну из задач из старинного учебника математики, который был написан в VII веке армянским ученым Анания Ширакаци.

Ответы на задание “Прилипшая монета” (начало его решения было приведено в рубрике “Крепкий орешек” в январском выпуске) представили:

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

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

Общероссийский проект

Школа цифрового века

Интернет-сопровождение проекта – Издательский дом «ПЕРВОЕ СЕНТЯБРЯ»

Общероссийский проект «Школа цифрового века» по комплексному обеспечению образовательных учреждений цифровыми предметно-методическими материалами разработан в соответствии с Федеральной целевой программой развития образования на 2011–2015 годы.

Проект направлен на развитие инновационного потенциала образовательных учреждений и вовлечение педагогических работников в цифровое образовательное пространство.

Открыт прием заявок от образовательных учреждений на 2013/14 учебный год

На новом этапе проекта

- Расширяется перечень предметно-методических журналов и дистанционных модульных курсов
- Реализуется специальная программа «Книги для учителя»
- Для образовательных учреждений, впервые входящих в проект, предусмотрена возможность получать материалы текущего 2012/13 учебного года

Оргвзнос от образовательного учреждения – 4 тысячи рублей за весь учебный год независимо от количества педагогических работников.

Участие образовательного учреждения и педагогических работников в проекте удостоверяется соответствующими документами. Для дошкольных учреждений предусмотрен свой набор удостоверяющих документов.

Срок действия проекта в 2013/14 учебном году: с 1 августа 2013 года по 30 июня 2014 года

Прием заявок и подробности

на сайте

digital.1september.ru

ИНФОРМАТИКА
Подписка: «Роспечать» – 32291 (бумажная версия), 19179 (электронная); «Почта России» – 79066 (бумажная версия), 12684 (электронная)