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

ИНФОРМАТИКА

внутри номера CD и код доступа к электронной версии

4 **Как много в этом звуке...** Цифр

34 **Не SQL'ем единым** Not only SQL

48 **Творения инженера Россума** и писателя Чапека

издательский дом 1september.ru **Первое сентября** | ИЮНЬ 2013

ИНФОРМАТИКА Подписка: «Роспечать» – 32291 (бумажная версия), 19179 (электронная); «Почта России» – 79066 (бумажная версия), 12684 (электронная)

НА ОБЛОЖКЕ

Лет 30 назад практически в каждом ВЦ на почетных местах можно было увидеть полотна, вышедшие из-под печатающих головок огромных матричных принтеров. Кто помнит те времена, знают, что не все, что висело тогда на стенах, можно использовать на обложке образовательного журнала, но некоторые изображения по праву могут считаться шедеврами ASCII-графики. Сейчас это искусство уже подзабыто, но до сих пор есть энтузиасты, создающие новые творения. Для этого даже имеются специализированные программы, хотя для эстетов жанра это не спортивно. Вот “ручками”...

В НОМЕРЕ

3 ПАРА СЛОВ
- Этого не может быть, потому что не может быть никогда

4 УГЛУБЛЕНКА
- Обработка звука
- Растровая графика
- Введение в SQL: практикум

34 СЕМИНАР
- Нереляционные базы данных: практикум

42 ТЕХНОЛОГИИ
- Информатика в облаках

48 ЗАНИМАТЕЛЬНЫЕ МАТЕРИАЛЫ ДЛЯ ПЫТЛИВЫХ УЧЕНИКОВ И ИХ ТАЛАНТЛИВЫХ УЧИТЕЛЕЙ
- “В мир информатики” № 188

НА ДИСКЕ

ЭЛЕКТРОННЫЕ МАТЕРИАЛЫ:

Исходные файлы и презентации к статьям номера

ИНФОРМАТИКА

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

http://inf.1september.ru

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

РЕДАКЦИЯ:
гл. редактор С.Л. Островский
редакторы
Е.В. Андреева,
Д.М. Златопольский
(редактор вкладки “В мир информатики”)
Дизайн макета И.Е. Лукьянов
верстка Н.И. Пронская
корректор Е.Л. Володина
секретарь Н.П. Медведева
Фото: фотобанк Shutterstock
Журнал распространяется по подписке
Цена свободная
Тираж 25 882 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 14.05.2013,
фактически 14.05.2013
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

“...Вы сочинили и напечатали в своем умном соченении, как сказал мне о. Герасим, что будто бы на самом величайшем светиле, на солнце, есть черные пятнушки. Этого не может быть, потому что этого не может быть никогда. Как Вы могли видеть на солнце пятны, если на солнце нельзя глядеть простыми человеческими глазами, и для чего на нем пятны, если и без них можно обойтиться? Из какого мокрого тела сделаны эти самые пятны, если они не сгорают? Может быть по-вашему и рыбы живут на солнце? Извините меня дурмана ядовитого, что так глупо сострил! Ужасно я предан науке!”

А.П. Чехов.
“Письмо к ученому соседу”

Этого не может быть, потому что не может быть никогда

Легкая и не драматическая зарисовка с хорошим концом

У моей коллеги (из замечательных) под конец учебного года случилась почти неожиданная радость. За майские праздники ей отремонтировали кабинет, причем очень быстро и очень качественно. Кабинет — действительно как с иголочки. Правда, случилось “но”. Электрики и компьютерщики сделали в кабинете новую разводку проводов так, как им показалось разумным и удобным. В результате учительское рабочее место теперь должно располагаться иначе по отношению к классу, не так, как было раньше. Я застал разговор коллеги с директором ее школы (директор — тоже замечательный, но имена на всякий случай изменены ☺).

— Анна Ивановна, ну не все ли Вам равно! Разницы ведь никакой!

— Мария Павловна, это неудачное место! Дети лучше усваивают новый материал, если я стою в этой части класса.

-— Ну, пожалуйста, не говорите ерунды. Вы — прекрасный учитель, дети у Вас усваивают материал, где бы Вы ни стояли.

— Я работала в разных классах. Я давно подметила эту закономерность, я считаю, что это важно, хотя сама не знаю почему.

— Анна Ивановна, давайте будем реалистами — разницы нет никакой. А новое место даже светлее. Если у Вас нет других аргументов…

— Ну, мне действительно так работать комфортнее! Я лучше себя чувствую на привычном месте.

— Хорошо. Я уверена, что все это капризы. Да и провода будут теперь видны. Но раз Вы так хотите…

— Очень хочу!

— Тогда перекладываем провода.

Правда, отличный директор? ☺ Мне было так радостно за коллегу и так приятно наблюдать за действиями директора, что решительно не хотелось вносить в эту идиллическую картину сухую нотку реализма и сообщать, что многолетние наблюдения Анны Ивановны являются экспериментальным подтверждением достаточно хорошо изученного механизма согласования режимов работы мозга с глазодвигательными реакциями. Ведь отлично, когда директору для принятия правильного решения совсем не надо знать всей этой “мути”. Но такие директора встречаются не часто (хотя и не очень редко). В иных же случаях иногда помогает Антон Павлович Чехов.

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

Обработка звука

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,

П.В. Бочаров,
учитель информатики, государственное бюджетное образовательное учреждение города Москвы Кадетская школа-интернат "Навигацкая школа"

Представление звука

Как известно из курса физики, звук — это воспринимаемые человеком колебания среды (воздуха или воды). Параметр, который фиксируется человеческим ухом, — это давление среды.

Как и любая волна, звук характеризуется двумя основными параметрами — **амплитудой** (т.е. величиной колебания) и **частотой** (т.е. количеством колебаний за единицу времени). Эти параметры в физических звуковых волнах постоянно изменяются, воспринимая их, мы слышим звук.

Поскольку в природе звуковая волна непрерывна, то для обработки звука с помощью компьютера необходимо построить модель его **оцифровки** — т.е. преобразования в числовое (двоичное) представление и обратное преобразование — для воспроизведения. Цифровое представление дискретно, поэтому процесс иногда называют *дискретизацией*.

Для выполнения оцифровки звука колебания давления среды с помощью микрофона переводятся в электрические колебания. Оцифровка звука выполняется как составление таблицы замеров возникающего напряжения. Замеры выполняются через определенные равные промежутки времени. Для того чтобы охарактеризовать этот промежуток, указывают **частоту дискретизации** — т.е. количество замеров за одну секунду. Точность оцифровки напрямую зависит от частоты — чем чаще замеры, тем точнее может быть передано колебание звука при воспроизведении. Если частота дискретизации ниже частоты колебаний самого звука, то изменения начинают теряться.

Второй параметр, от которого зависит точность воспроизведения, — это

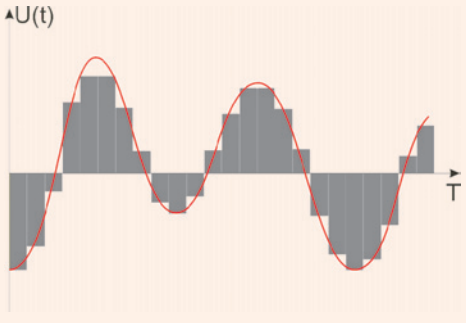


Рис. 1. Оцифровка звука

Рис. 2

количество возможных градаций напряжения. Это количество градаций зависит от количества битов, отведенных на запись значения, т.е. разрешения (обычно 24 бита).

При профессиональном обсуждении каждое измерение давления будет называться *отсчетом*. Чем выше частота отсчетов и чем больше глубина кодирования, тем естественнее и качественнее будет компьютерный звук.

Как и в случае с графическими данными, представление звука в виде потока замеров порождает большой объем данных, затрудняющий и хранение и передачу звука.

При прямой записи объем получающихся данных достаточно велик. Например, при записи 16-битного звука с частотой 22 КГц одна минута звучания займет в памяти 60 секунд * 22 000 значений * 2 байта = 2578 Кб 128 байт.

Объем данных для приведенного рисунка вы можете подсчитать сами.

Для решения проблем, связанных с большим объемом, при работе со звуком часто используют методы сжатия с потерями, использующими особенности восприятия звука человеком, которые изучаются специальной научной дисциплиной — психоакустикой.

Особенную популярность при упаковке звука завоевал стандарт сжатия **MPEG-1 Layer 3** (сокращенно называется **MP3**[1]), разработанный Институтом Фраунхоффера. При сжатии этим методом используется сжатие с потерей информации. При этом учитывается особенность слухового восприятия: если рядом расположены две частоты, то более громкая “перекрывает” более тихую. Таким образом, ее можно сгладить без ощутимой потери качества звука. Для реализации этого эффекта весь поток разделяется на равные небольшие фрагменты, в которых и выполняется удаление незаметных человеческим ухом звуков.

[1] Существуют и MPEG-2 Layer 3, и MPEG-2.5 Layer 3.

Формат MP3 позволяет хранить моно- или стереозвук и таким образом для высококачественного кинозвука не очень подходит.

Основным параметром, определяющим качество звучания, для формата mp3 будет ***битрейт*** (*Bit Rate* — битовая скорость), то есть количество битов в закодированном потоке в секунду. Считается, что человеческое ухо не различает изменения в звуке при битрейте 128 КБит/сек., но на практике это зависит и от средств кодирования (MP3 — не точный алгоритм, а формат хранения), и от средств воспроизведения. Формат MP3 позволяет работать с битрейтами до 320 Кбит/сек., причем битрейт может быть как постоянным для всего потока, так и переменным.

На практике даже опытный эксперт часто не фиксирует разницы при битрейте 256 Кбит/сек.

Кроме сжатых звуковых данных, формат MP3 позволяет хранить и информацию о звуковом фрагменте — название, автора и т.п. с помощью так называемых “тегов”.

Помимо обработки звука как последовательности замеров давления, очень часто компьютер используется как средство создания мелодии, моделируя звучание разных инструментов.

Мелодия синтезируется как проигрывание нот каким-либо инструментом, который ее должен исполнять. Поскольку сложные мелодии исполняются согласованно несколькими инструментами (говорят — голосами), то современные программы такого рода обеспечивают работу с несколькими **голосами**.

Ноты в данном случае выступают как основные, несущие звуки. Частоты нот (в герцах) известны (см. таблицу на с. 6).

Если точно воспроизвести ноты по этим частотам без добавлений, то мелодия будет похожа на мелодию флейты.

Для синтеза звука с другими инструментами применяются средства, получившие названия ап-

Нота	Частота, Гц								
	Суб-контр-октава	Контр-октава	Большая октава	Малая октава	1.00 октава	2.00 октава	3.00 октава	4.00 октава	5.00 октава
До		32.70	65.41	130.82	261.63	523.25	1046.50	2093.00	4186.00
До-диез		34.65	69.30	138.59	277.18	554.36	1108.70	2217.40	4434.80
Ре		36.95	73.91	147.83	293.66	587.32	1174.60	2349.20	4698.40
Ре-диез		38.88	77.78	155.56	311.13	622.26	1244.50	2489.00	4978.00
Ми	20.61	41.21	82.41	164.81	329.63	659.26	1318.50	2637.00	5274.00
Фа	21.82	43.65	87.31	174.62	349.23	698.46	1396.90	2793.80	
Фа-диез	23.12	46.25	92.50	185.00	369.99	739.98	1480.00	2960.00	
Соль	24.50	49.00	98.00	196.00	392.00	784.00	1568.00	3136.00	
Соль-диез	25.95	51.90	103.80	207.00	415.30	830.60	1661.20	3332.40	
Ля	27.50	55.00	110.00	220.00	440.00	880.00	1720.00	3440.00	
Си-бемоль	29.13	58.26	116.54	233.08	466.16	932.32	1864.60	3729.20	
Си	30.87	61.74	123.48	246.96	493.88	987.75	1975.50	3951.00	

паратных и программных **секвенсоров** (синтезаторов). Такое средство синтезирует звук на основе указания тона (ноты), времени ее звучания и образца звучания того или иного инструмента. Образец звучания называется **сэмплом**.

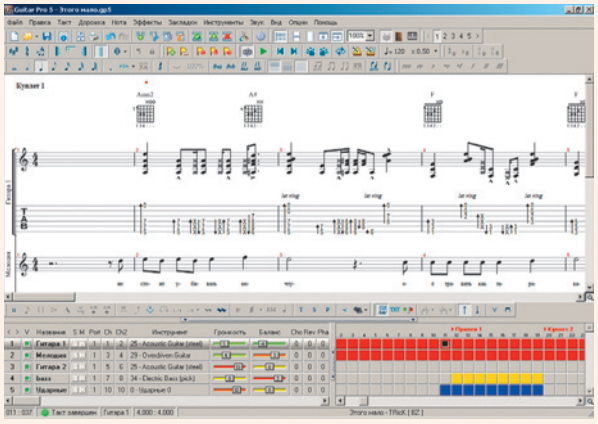


Рис. 3. Подготовка музыки в программе Guitar Pro

Для обеспечения качественного звучания большого количества разнообразных инструментов разрабатываются специальные библиотеки сэмплов. Современные звуковые карты обладают собственной памятью, позволяющей загружать такие библиотеки от сторонних разработчиков и процессорами, выполняющими синтез звука.

При хранении и обмене синтезируемая мелодия описывается стандартизированным набором команд, получившим название **MIDI** (*Musical Instrument Digital Interface*). Стандарт включает в себя протоколы взаимодействия аппаратных средств (например, звуковой карты и отдельного устройства-синтезатора) и описание формата хранения файлов.

Для ввода мелодии в стандарте MIDI в компьютер применяется специальное средство ввода — **MIDI-клавиатура**. Фактически она похожа на клавиатуру пианино, но предназначена для передачи соответствующих нот звуковой карте — для последующего синтеза звука заданным инструментом. Программы синтеза мелодий отображают набранные ноты в соответствии с музыкальной нотацией и позволяют редактировать их.

В этом практикуме мы будем говорить прежде всего об обработке музыкальных файлов, представленных в виде оцифрованного звука, поскольку подготовка MIDI-записи требует наличия музыкального образования и уверенного владения нотной записью.

Практикум по обработке звука состоит из 10 заданий-этапов, пройдя которые мы получим готовую к тиражированию музыкальную композицию. Само собой, чтобы выполнить эту часть работы звукорежиссера, требуется записанный звук, который вы можете найти на диске.

Задание 1. Основы звукорежиссуры

Программное обеспечение

На сегодняшний день компьютер позволяет знающему человеку делать со звуком самые невероятные вещи.

Программ обработки звука огромное количество. Некоторые из них позволяют изменить скорость воспроизведения звукового файла, другие — изменить тональность записанной музыки и многое-многое другое. Программы, позволяющие выполнить большинство операций с музыкальными файлами (фактически — последовательностями звуков), называются **музыкальными секвенсорами** и предназначены для профессиональной звукорежиссерской работы.

К таким программам относятся *Adobe Audition, Fruity Loops Studio, Cubase, Nuendo* и многие другие. Более простые для освоения и работы, на взгляд авторов, продукты компании *Steinberg* — *Cubase* и *Nuendo*. Обе эти программы имеют схожий интерфейс и незначительные отличия. В этом практикуме мы будем осуществлять обработку звука в программе *Nuendo 3*.

По типу использования *Nuendo* относится к условно-бесплатному типу программного обеспечения. Программой можно бесплатно пользоваться в течение нескольких сот рабочих часов. Этого хватит для выполнения заданий практикума. Скачать пробную версию можно бесплатно с сайта разработчика *www.steinberg.net*.

Процесс обработки записанной музыки

Процесс обработки записанной музыки в виде отдельных файлов (дорожек или треков), каждый из которых может содержать записанные либо партии музыкального инструмента, либо вокала, либо просто некоторые звуки, принято делить на три части:

1. ***Трекинг*** (*Tracking*) — процесс редактирования записанных звуковых файлов с помощью стандартных инструментов секвенсора (обрезка, склейка и прочее).
2. ***Сведение*** (*Mixing*) — этап создания единого звукового трека, на котором основная цель звукорежиссера — добиться, чтобы каждая из записанных дорожек была отчетливо слышна в общем звучании всех треков.
3. ***Мастеринг*** (*Mastering*) — финальный этап работы со сведенной звуковой дорожкой. На этом этапе основной задачей звукорежиссера является добиться максимально качественного звучания в рамках стиля музыкального трека, а также устранить оставшиеся недостатки звука в целом.

Задание 2. Интерфейс программы Nuendo

Как уже говорилось, программа *Nuendo* является многофункциональным музыкальным секвенсором. Мы изучим лишь основные ее возможности.

2.1. Запустите программу.

2.2. При первом запуске программа предложит протестировать имеющееся звуковое оборудование — согласитесь.

2.3. Откроется окно программы:

Рис. 4

2.4. Изучите элементы меню программы, которые нам понадобятся:

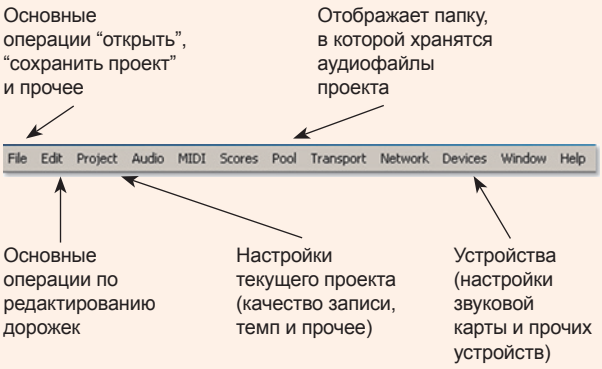


Рис. 5

2.5. Проверьте, какой звуковой драйвер выбран в настройках программы. Если выбран один из стандартных драйверов производителя секвенсора — поменяйте его на драйвер своей звуковой карты. Если этого не сделать, программа, возможно, будет работать некорректно. Выбирая драйвер карты, вы сообщите программе обработки целый ряд параметров, которые обычно не учитывают — поскольку обработкой звука как таковой не занимаются.

Выбор драйвера своей звуковой карты:

Меню → Devices → Device Setup

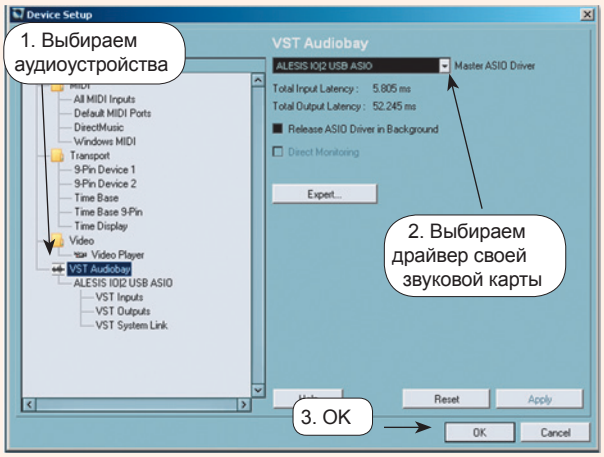


Рис. 6

Если у вас возникают трудности с выбором драйвера — скачайте из Интернета и установите драйвер *ASIO4ALL*. Этот драйвер универсален и должен работать со всеми звуковыми картами.

Задание 3. Создание и настройка проекта

Для того чтобы начать обработку треков, требуется создать проект с определенными настройками и разместить треки в проекте.

3.1. Создайте новый проект:

Меню → File → New Project

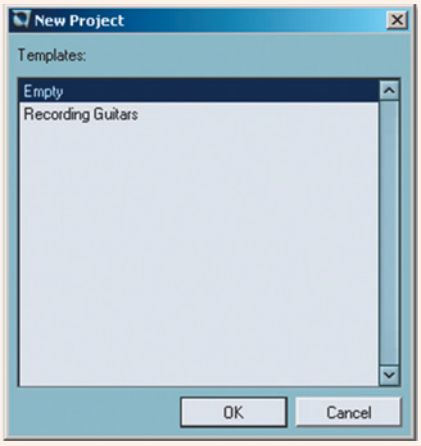


Рис. 7

3.2. Здесь нам предлагается выбрать ранее сохраненные настройки проекта. Поскольку мы создаем проект впервые — выбираем *Empty* (пустой).

3.3. Перед нами дерево каталогов нашего компьютера, где мы должны выбрать папку, в которой будет храниться наш проект. Чтобы не превращать все в свалку, лучше всего для проектов создать отдельную папку и помещать все папки с дальнейшими проектами туда. Итак, создайте такую папку, назовем ее, к примеру, *Projects*, в ней создайте папку *First Project*. Выбираем эту папку и нажимаем *OK*. В папке *First Project* у нас автоматически будет создана папка *Audio*, в этой папке будут храниться все звуковые файлы проекта.

3.4. Перед нами появится окно проекта. Ниже представлен снимок проекта с уже загруженными аудиодорожками.

Изучите основные области окна проекта (*рис.* 8).

3.5. Откройте окно настроек проекта:

Меню → Project → Project Setup

3.6. Настройте параметры проекта:

Sample Rate — **44 100 Hz** (частота отсчетов в секунду при кодировании звука);

Record format — **24 Bit** (глубина кодирования — количество бит для кодирования каждого отсчета);

Record File Type — **Wave** (разрешение/тип аудиофайлов проекта).

Указанные настройки являются оптимальными, и большинство используют именно их. Можно указать настройки и выше, но изменения в качестве вряд ли будут различимы, зато размер звукового файла значительно увеличится.

3.7. Сохраните ваш проект.

Меню → File → Save As

Укажите имя файла, например, *Tracking and Mixing*. После сохранения в папке проекта появится файл проекта, с которым мы можем продолжить работу в любое время.

Задание 4. Добавление аудио в проект

4.1. Сейчас в нашем проекте нет ни одного звукового файла. Добавим их:

Pool → Open Pool Window

Правой кнопкой мыши нажимаем на иконке *Audio → Import Medium*.

4.2. Через проводник найдите в папке практикума “*Обработка звука*” папку “*Instruments*”, в ней

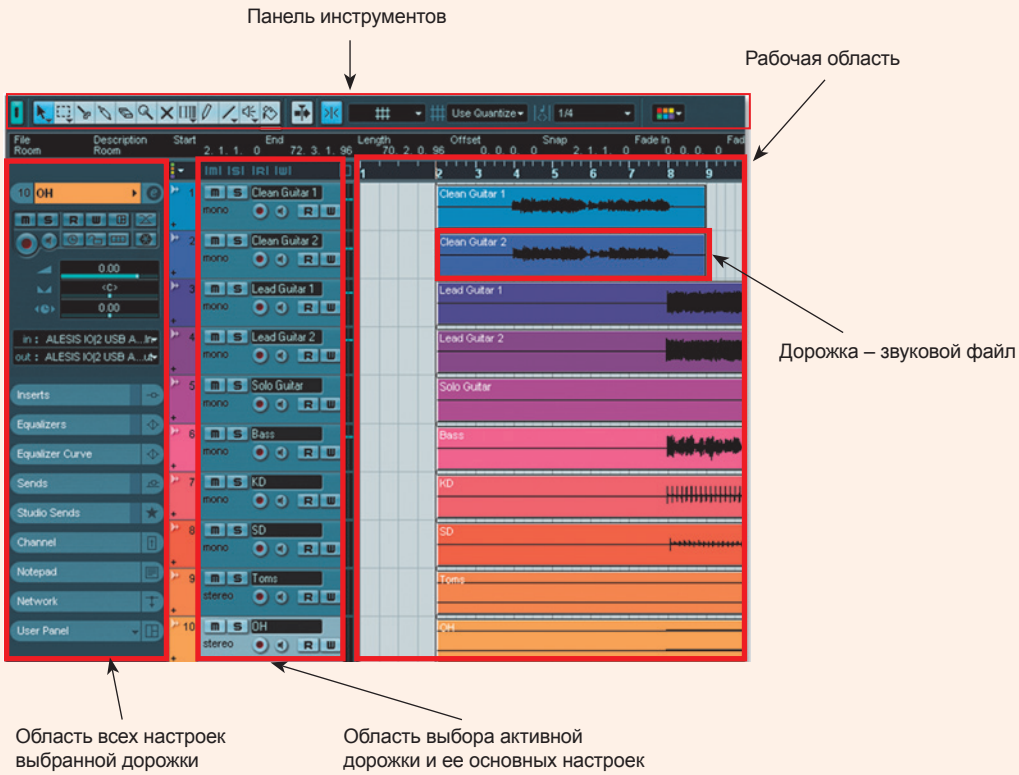


Рис. 8

лежат записанные партии инструментов к песне, с которой мы будем работать.

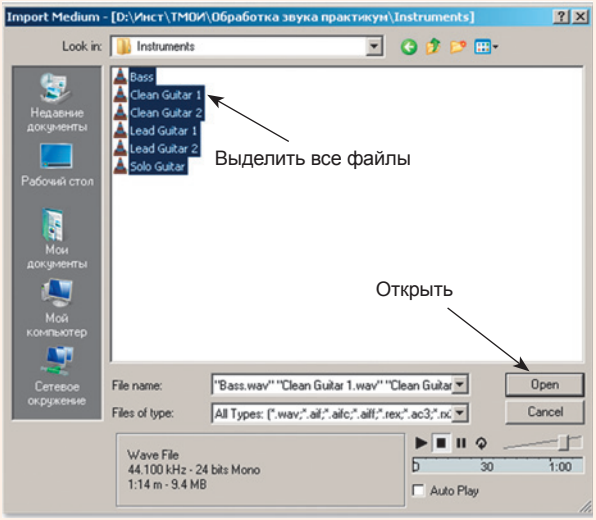


Рис. 9

4.3. В появившемся окне импорта отметьте указанные пункты:

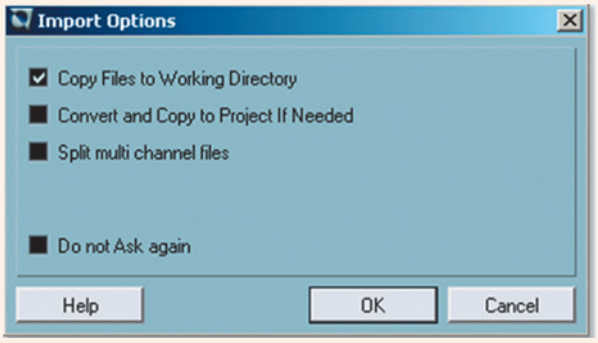


Рис. 10

Нам нужно выбрать первый пункт — скопировать файлы в папку проекта, если этого не сделать — никакие дальнейшие операции с файлами, кроме прослушивания, выполнять будет нельзя.

Второй пункт предлагает при копировании в папку проекта конвертировать копируемые файлы в файлы с расширением проекта и его параметрами, чтобы с ними можно было работать. Этот пункт мы можем не выбирать, так как все эти файлы уже в формате *Wave* и имеют те же параметры, что и наш проект.

Разделение многоканальных (стерео и пр.) файлов нам тоже не нужно. Пока мы будем работать только с монодорожками.

Нажимаем *OK*.

В результате мы загрузили в проект все необходимые партии — то есть дорожки, из которых он будет состоять.

4.4. Разместим загруженные дорожки в проекте.

Закройте *Pool Window* и нажмите правой кнопкой на пока еще пустую область всех дорожек (см. *рис.* 10).

Появится список всех типов треков, которые мы можем создать. Нам нужно добавить аудиотрек — нажимаем *Add Audio Track*.

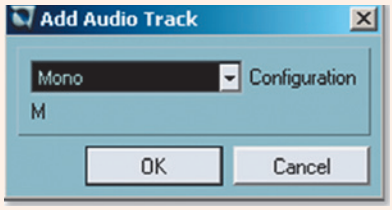


Рис. 11

Появилось окно выбора конфигурации трека (сколько каналов будет в треке) — выбираем *Mono*.

4.5. Вместо названия *Audio 01* напишите более понятное название появившегося трека — *Clean Guitar 1*:

Рис. 12

4.6. Дальше поместим соответствующий звуковой файл из *Pool Window* в рабочую область на эту дорожку.

Открываем *Pool Window*, перетаскиваем мышкой файл *Clean Guitar 1* на рабочую область соответствующей дорожки. Должно получиться так:

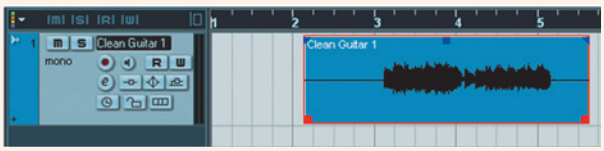


Рис. 13

Сверху над изображением звукового файла у нас идут цифры 1, 2, ... — это такты сетки метронома проекта. При работе с музыкой удобнее всего работать именно с ней. Если ее нет — ее можно настроить в *Project Setup* (настройках проекта).

4.7. Поставьте курсор проигрывания перед дорожкой:

Рис. 14

Для этого кликните на поле сетки метронома в нижней его части:

Рис. 15

Нажмите клавишу “пробел” для воспроизведения дорожки с места, где установлен курсор.

4.8. Теперь добавим еще один файл в поле проекта, чтобы можно было одновременно слушать две гитары.

Создайте еще одну звуковую дорожку, как мы это только что сделали, и добавьте в нее файл *Clean Guitar 2*. Должно получиться примерно так:

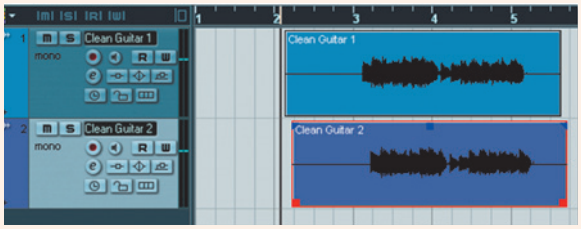


Рис. 16

Воспроизведите. Две гитары звучат одновременно. Правда, мы их еще не выровняли, поэтому они могут немного “не попадать” друг в друга.

4.9. Чтобы не выравнивать дорожки вручную, в *Nuendo* есть возможность притягивать их к общей сетке метронома. Именно так на записи добиваются идеального попадания музыкантов в ритм. Чтобы это можно было сделать, установите следующие настройки на панели инструментов:

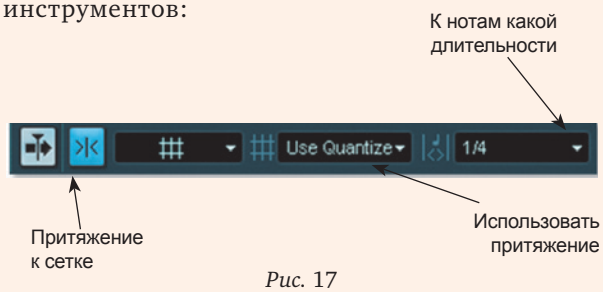


Рис. 17

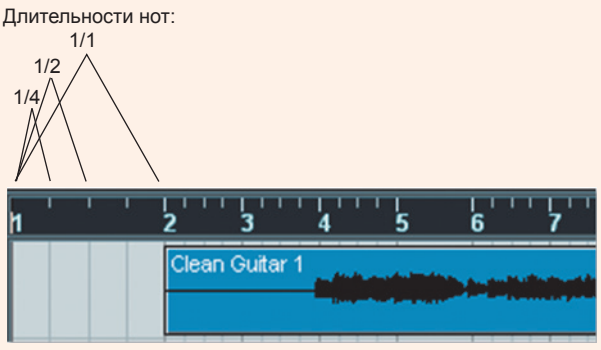


Рис. 18

Если какого-то элемента нет на панели инструментов, то можно нажать на ней правой кнопкой мыши и отметить галочкой необходимый инструмент:

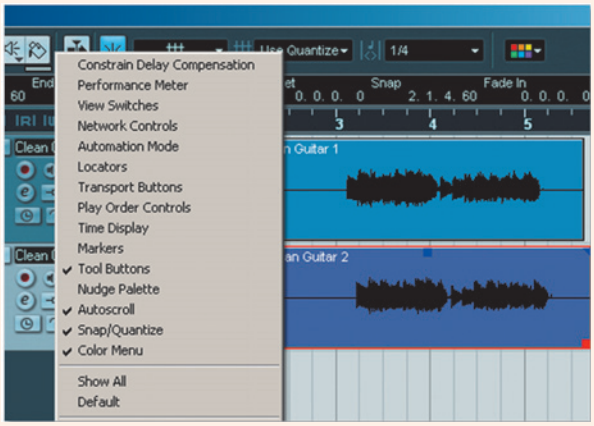


Рис. 19

4.10. После включения притяжения к сетке, выбрав на панели инструмент “*Перемещение*”  , можно двигать дорожки четко по четвертым нотам сетки. Сделайте так, чтобы файлы начинались воспроизводиться точно в одном месте — с начала второго такта:

Рис. 20

Воспроизведите — теперь гитары должны “попадать” друг в друга.

4.11. Добавьте аналогичным образом остальные файлы из *Pool Window* в проект. Сделайте так, чтобы дорожки все начинали воспроизводиться с начала второго такта. Должно получиться так:

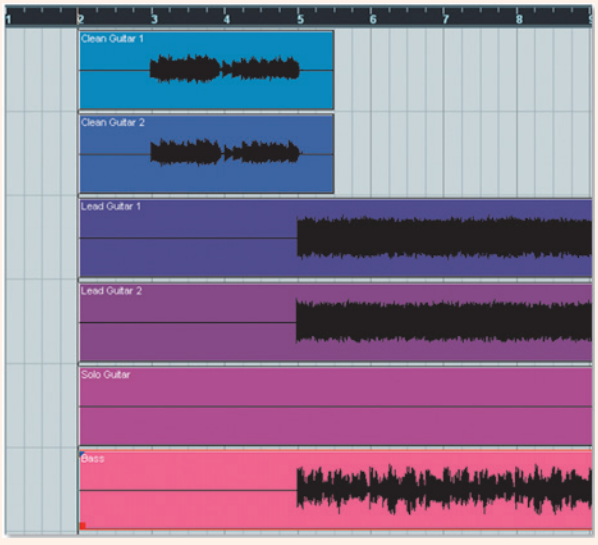


Рис. 21

4.12. Воспроизведите и послушайте результат.

Задание 5. Трекинг

5.1. Сейчас мы занимаемся первым этапом редактирования — трекингом. Здесь в задачу звукорежиссера входит сделать так, чтобы каждый трек звучал идеально для дальнейшего сведения. Послушаем отдельно дорожку с бас-гитарой. Нажмем кнопку S (*Solo*) на дорожке *Bass*.

5.2. Воспроизведите, послушайте. Слышите щелчки при склейке кусков? Это абсолютно нормальное явление при склейке на записи. И это очень просто убрать с помощью *Nuendo*.

5.3. Щелчки находятся на сетке метронома. Так как в середине песни темп (скорость игры музыкантов) меняется, нам нужно загрузить темпотрек (темп, отображающийся в виде общей сетки метронома), данный для этой песни:

Меню → File → Import → Tempo Track

Выбираем файл *tempo_track* в папке практикума. Теперь все места склейки находятся на линиях сетки метронома.

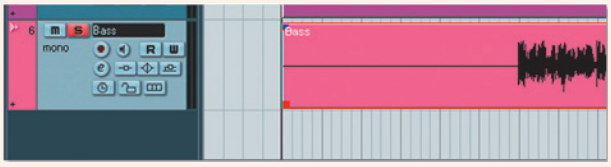


Рис. 22

5.4. Теперь слушаем, где щелчки у баса. Между *11-м* и *12-м* тактами есть щелчок. Выберите инструмент *"Ножницы"* на панели инструментов и разрежьте файл между *11-м* и *12-м* тактами (притяжение к сетке должно быть включено — иначе разрезать дорожку точно по сетке сложно). Выделите две полученные дорожки инструментом с зажатой клавишей Shift:

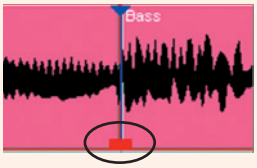


Рис. 23

5.5. Сделайте склейку выделенных треков, нажав клавишу *X* на клавиатуре.

Результат:

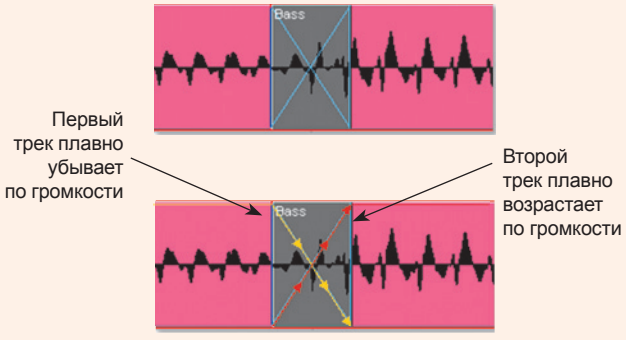


Рис. 24

Воспроизведите. Щелчок должен убраться. Сделайте так с остальной частью басовой дорожки, где услышите щелчки (ищите на началах такта).

Выполненная операция называется *Crossfade* — плавный переход уровня сигнала.

5.6. Добавим в проект ударные инструменты. В *Pool Window* добавим для удобства отдельную папку барабанов в папке *Audio*. Правой кнопкой нажимаем на папке *Audio* → *Create Folder*. Появится новая папка — переименуйте ее в *Drums* (барабаны). Добавьте туда все файлы из папки *Drums* в папке практикума (правой кнопкой мышки в *Pool Window* на папке *Drums*, дальше аналогично).

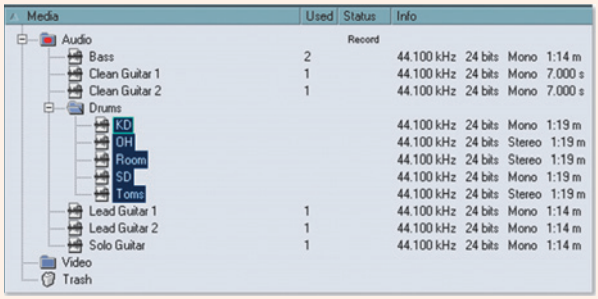


Рис. 25

Должно получиться, как показано на *рис.* 25. Справа всегда указывается информация о файлах. Заметьте, из новых файлов три имеют конфигурацию *Stereo*, такую конфигурацию нужно будет выбрать и у создаваемого трека для этого файла.

Поместите партии ударных инструментов в проекте аналогично инструментам. Прировняйте все партии к началу второго такта:

Рис. 26

Задание 6. Сведение, панорамирование

Итак, у нас теперь каждая партия находится в проекте, все дорожки выровнены и склеены. Теперь мы можем приступать к сведению. Для начала прослушаем еще раз то, что у нас уже есть без какой-либо обработки. Звучит не очень, правда?

Громкости не настроены, что-то теряется, что-то слишком выбивается, все инструменты перекрывают друг друга. Ликвидировать все эти недоразумения и сделать общее звучание более выразительным мы можем на этапе сведения. Вот основные его шаги:

1. *Панорамирование*.
2. *Эквализация*.
3. *Компрессия*.
4. *Дополнительные эффекты*.

Разберемся с каждым из них подробнее.

Панорамирование

Панорамирование — это первый этап сведения, на котором каждой дорожке выделяется отдельная область в пространстве.

У нас две колонки и из них играет музыка, о каком пространстве идет речь?

Все не так просто. Стереозвучание, как вы, должно быть, знаете, не для того, чтобы звук из двух колонок был просто громче. Если вы, например, включите свою любимую группу на плеере и послушаете музыку через один наушник — вы все поймете.

Дело в том, что при сведении музыки, например, одной гитаре выделяется полностью правый канал (колонка, наушник), а другой — полностью левый. Таким образом, создается имитация объема, с разных сторон которого звучат разные музыкальные инструменты (создается панорама). У слушателя возникает ощущение, что справа от него играет один инструмент, а слева — другой. При этом инструменты, "разведенные" по панораме, уже не будут так перекрывать друг друга.

В музыке часто можно услышать, как какой-то звук как бы "перетекает" из одного наушника в другой, это тоже достигается за счет панорамы.

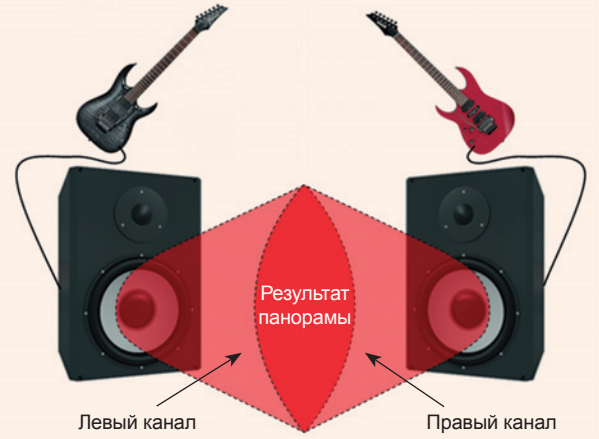


Рис. 27

6.1. Итак, "разведем" гитары по панораме. Для этого откроем микшерный пульт (микшер) *Nuendo*:

Меню → *Devices* → *Mixer* (или клавиша F3)

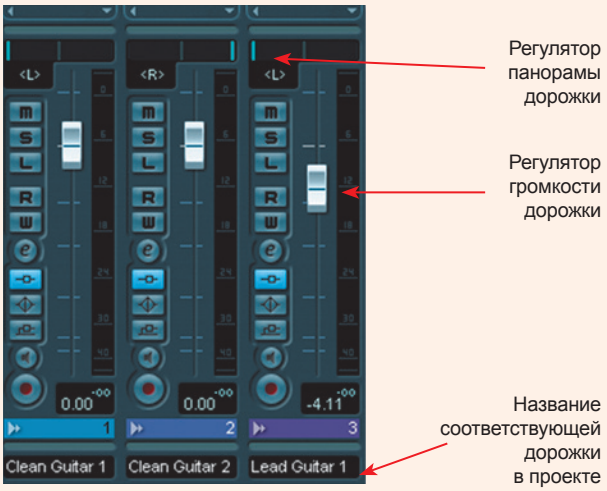


Рис. 28

Нажмите на микшере кнопку S (*Solo*) на дорожках *Lead Guitar 1* и *Lead Guitar 2*, чтобы слышать только их.

С помощью регуляторов панорамы поместите дорожку *Lead Guitar 1* полностью в левый канал, а *Lead Guitar 2* — полностью в правый канал. Прослушайте результат. Улучшилось ли звучание?

Заметьте, что дорожка может находиться не полностью в каком-то одном канале, а, например, только на 80% в левом. Это значит, что 20% останутся в правом канале и мы их будем немного слышать правым ухом. Таким образом, у нас возникнет ощущение, что звук играет не слева, а слева спереди. Это особенно заметно, если слушать музыку в наушниках.

6.2. Аналогично "разведите" по панораме дорожки *Clean Guitar 1* и *Clean Guitar 2*. Прослушайте результат. Общее звучание должно стать лучше, гитары не будут наслаиваться друг на друга.

У остальных дорожек не меняйте панораму, все монодорожки будут по умолчанию в центре панорамы, а стереодорожки уже разведены по панораме (послушайте отдельно стереодорожку *OH*).

6.3. С помощью регуляторов громкости убавьте звук у дорожек гитар и баса, так чтобы они не заглушали барабаны и мы могли по возможности слышать каждую дорожку.

Задание 7. Сведение, эквализация

Эквализация

Эквализация — это второй и один из самых важных этапов сведения. Что же такое эквализация? Слово кажется сложным, но на самом деле все элементарно.

Звуки можно разделить на звуки низкие по частоте, средние и высокие. Например, бас-гитара имеет низкое звучание — по большей части ее звук находится в низких частотах и занимает их почти целиком. Обычная гитара занимает большинство средних частот и часть высоких. "Железо" барабанов (тарелки) — большинство высоких.

Однако бас-гитара все равно немного звучит в области средних и в области высоких частот, тем самым перекрывая звучание обычных гитар и тарелок. Так и любой другой записанный инструмент все равно немного будет звучать вне своего основного частотного диапазона.

Эквализация помогает убавить громкость ненужных частот у звука и поднять громкость тех частот, на которых инструмент будет хорошо звучать вместе с остальными. Попробуем это проделать.

Подобная обработка ведется с помощью расширений программы — так называемых **звуковых плагинов**, которые реализует большинство эффектов при обработке. Наложим нужные эффекты на отдельные треки.

7.1. Откройте миксер. Добавьте расширенный миксер и в нем выберите *Разрывы* (поля для вставки звуковых плагинов) всем дорожкам:

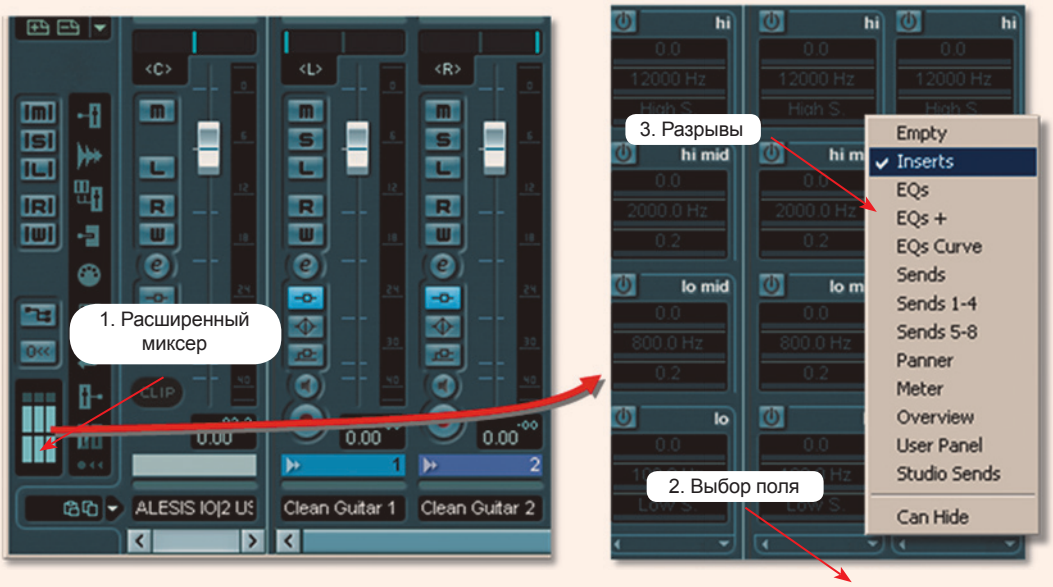


Рис. 29

7.2. Вставьте эквалайзер в первый *Разрыв* трека *Clean Guitar 1*.

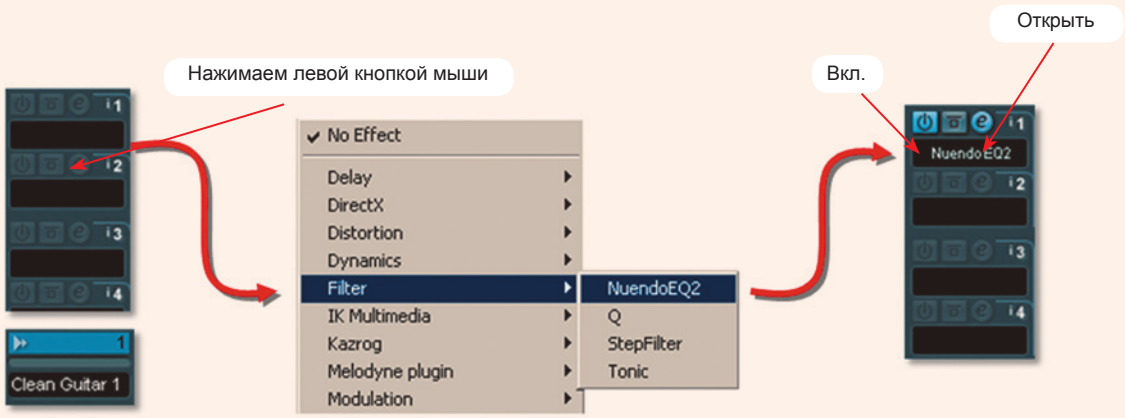


Рис. 30

7.3. Установите режим *Solo* только дорожке *Clean Guitar 1*, чтобы услышать, как эквалайзер меняет звук:

Рис. 31

7.4. Откройте и изучите окно эквалайзера. Изучите, как настроить эквалайзер:

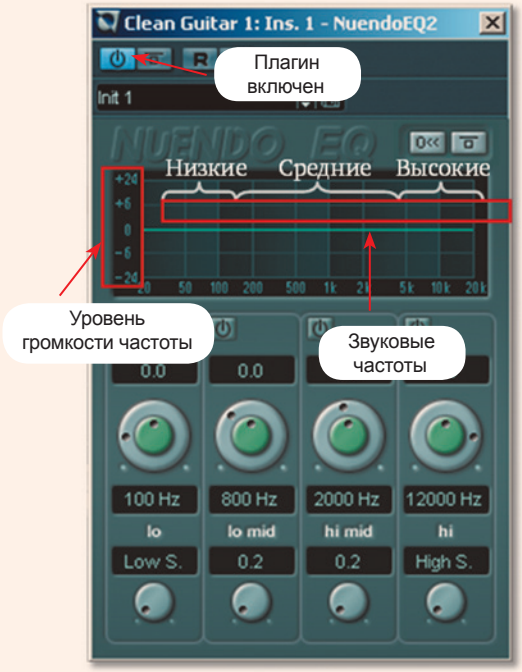


Рис. 32

Рис. 33

Поэкспериментируйте с ручками, чтобы лучше разобраться, как изменение их положения влияет на кривую эквалайзера и на звук дорожки.

7.5. Частотный диапазон гитары обычно выше *150 Герц*. Убавьте у дорожек *Clean Guitar 1, Clean Guitar 2, Lead Guitar 1, Lead Guitar 2* низкие частоты (см. *рис.* 34).

7.6. Сделайте эквализацию у других дорожек:

1. ***Solo Guitar*** — убавить низкие частоты до *200 Hz* и "узко" поднять *5000 Hz (5 kHz)* на *10* децибел (*дБ*).

Рис. 34

2. ***Bass Guitar*** — убавить низкие частоты до *50 Hz* и "узко" поднять *150 Hz* и *5 kHz* на *10 дБ*, "узко" понизить *300 Hz* на *10 дБ*.

3. ***KD*** — убавить низкие частоты до *60 Hz*, "узко" поднять *80 Hz* и *5 kHz* на *10 дБ*.

4. ***SD*** — убавить низкие частоты до *100 Hz*, "узко" поднять *200 Hz* и *5 kHz* на *10 дБ*.

5. ***Toms*** — убавить низкие частоты до *100 Hz*, "узко" поднять *200 Hz* и *5 kHz* на *10 дБ*.

6. ***OH*** — убавить низкие частоты до *550 Hz*.

7. ***Room*** — ничего не меняем.

7.7. Настройки эквалайзера могут варьироваться на вкус звукорежиссера, так как каждый проект уникален, у разных инструментов немного различаются частотные диапазоны.

Определить точно, в каком частотном диапазоне звучит конкретный инструмент, могут помочь более многофункциональные эквалайзеры, которые при воспроизведении трека отображают уровень звучания инструмента на каждой частотной полосе.

Воспроизведите то, что у нас получилось. Для сравнения можно на время отключить все эквалайзеры, которые мы загрузили, и послушать, какой звук был раньше.

При обработке звука очень полезно делать такие сравнения. Включите все эквалайзеры снова. Эквализация завершена.

Задание 8. Сведение, компрессия

Компрессия

Компрессия — это третий шаг в сведении композиции. Что такое компрессия?

Сравните, например, дорожки *Lead Guitar 1* и *Bass*:

Рис. 35

Дорожка *Lead Guitar* по рисунку уровня громкости ровная на протяжении всего трека. Дорожка *Bass*, наоборот, динамична на протяжении всего трека.

Динамика в игре на музыкальном инструменте — естественное явление, так как музыкант не робот и не может играть все ноты с одинаковой громкостью. Однако в общем звучании такая динамика иногда очень сильно “режет слух”. Поэтому мы ее уберем компрессором (плагином, который в заданной степени уменьшает громкость звуков, уровень громкости которых выше указанного порога). На *рис.* 36 представлен отрывок дорожки баса не компрессированный, компрессированный и обработанный лимитером (плагином, выполняющим полное “выравнивание” всех нот по громкости):

Рис. 36

8.1. Добавьте компрессор во второй разрыв дорожки *Bass* и изучите его окно:

8.2. Настроим границу срабатывания *Threshold*, чтобы “сгладить” пики — то есть уменьшить их громкость на *10 дБ*.

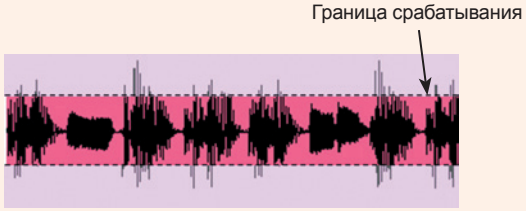


Рис. 38

Включите проигрывание дорожки баса. Меняйте границу срабатывания и следите на компрессоре за шкалой *Gain Reduction* (сколько *дБ* “сглаживается”). Выберите такую границу, чтобы показатель “сглаживания” *Gain Reduction* колебался в области *10 дБ*.

8.3. Установите максимальную степень компрессии.

Установите минимальное значение скорости срабатывания (*Attack*) и скорость выключения (*Release*) оставьте по умолчанию.

8.4. Поднимите выровненный звук на 10 “сглаженных” децибелов (*Make Up Gain*). Прослушайте результат. Отключите компрессор и сравните с предыдущим звучанием.

8.5. Включите режим лимитер в окне плагина (см. *рис.* 39).

Сделаем наш бас идеально ровным по громкости.

Выставите границу срабатывания *–12 дБ*. Таким образом, лимитер будет “срезать” все звуки, уровень громкости которых выше *–12 дБ*.

Послушайте результат.

Рис. 39

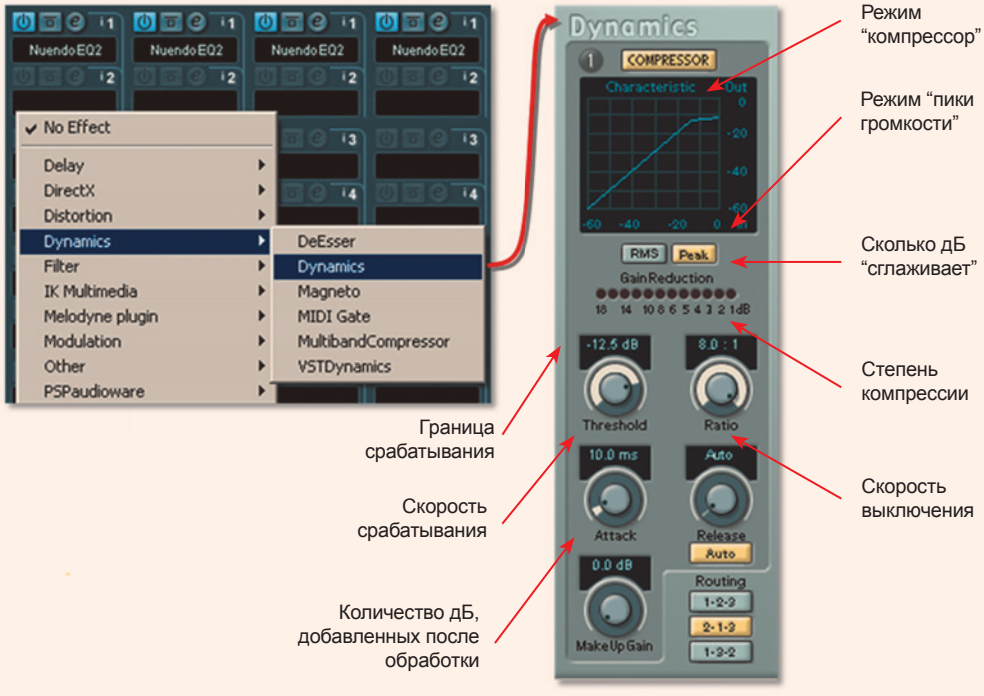


Рис. 37

8.6. Сделайте компрессию дорожек:

1. ***KD*** — Алгоритм — *Peak, Trashold — (–32) дБ, Ratio: 8:1, Attack — 0.1, Release — Auto, Make Up Gain — 15 дб.*

2. ***SD*** — Алгоритм — *Peak, Trashold — (–40) дБ, Ratio: 8:1, Attack — 10, Release — Auto, Make Up Gain — 10 дб.*

3. ***Toms*** — Так же, как у дорожки *SD*.

У компрессии много других применений, например, можно сделать несколько компрессий и звук станет довольно специфичным. У каждого звукорежиссера есть свои секреты по использованию компрессоров.

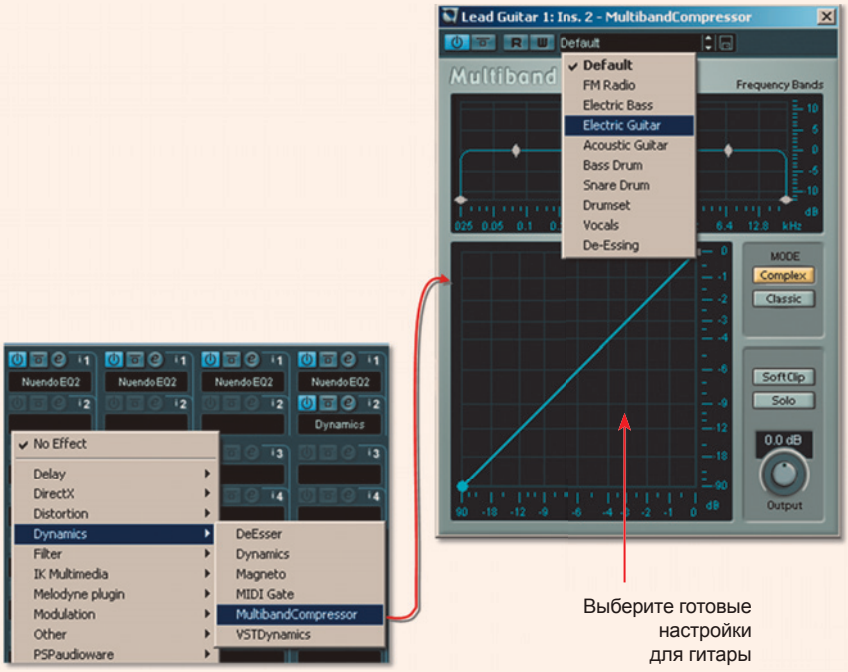


Рис. 40

Задание 9. Сведение, дополнительные эффекты

Дополнительные эффекты

Обычно трех перечисленных выше этапов профессиональному звукорежиссеру будет достаточно, для того чтобы свести трек на хорошем уровне. Но, как правило, появляется желание добавить еще что-нибудь. Здесь уже все зависит от фантазии и знаний звукорежиссера. Расскажем о двух часто применяемых эффектах.

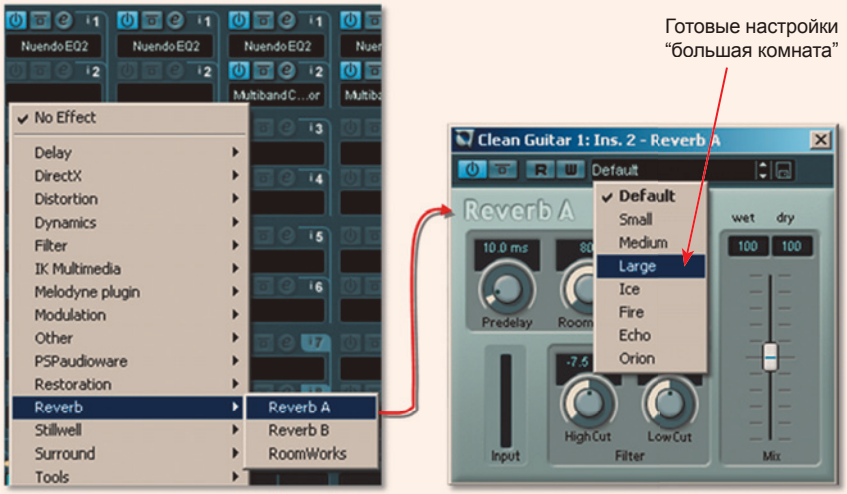


Рис. 41

Мультиполосная компрессия. Вкратце, это компрессия определенных частотных диапазонов. Этот эффект может придать инструменту более яркое звучание. Сделаем это для гитар в нашем проекте.

9.1. Откройте микшер и установите дорожкам *Lead Guitar 1* и *Lead Guitar 2* режим *Solo*.

9.2. Во вторые разрывы треков *Lead Guitar 1* и *Lead Guitar 2* установите плагин (см. *рис*. 40).

9.3. Послушайте результат. Выключите мультиполосную компрессию и сравните с тем, что было ранее.

9.4. Сделайте мультиполосную компрессию для дорожек: *Solo Guitar, Bass, KD, SD* и *Toms* с соответствующими готовыми настройками: *Electric Guitar, Electric Bass, Bass Drum, Snare Drum* и *Snare Drum*.

9.5. Послушайте результат. Выключите мультиполосную компрессию и сравните с тем, что было ранее.

Реверберация. Реверберацией называют эффект, при котором звук прекращает свое звучание не сразу, а продолжает некоторое время отдаленно звучать в пространстве. Представьте, что вы находитесь в длинном коридоре с гладкими твердыми стенами. Издавая какие-либо звуки, вы можете слышать, как они распространяются вдаль по коридору, отражаясь от стен. Это и есть эффект реверберации. Применение его к некоторым трекам делает их звучание лучше. Этот эффект обычно не применяется к басу и другим дорожкам, которые "поддерживают" ритм композиции. Обычно этот эффект применяется к вокалу, соло-гитаре, к акустическим гитарам. В нашем проекте вокала нет, поэтому сделаем реверберацию на дорожках *Clean Guitar 1* и *Clean Guitar 2*.

9.6. Во второй разрыв треков *Clean Guitar 1* и *Clean Guitar 2* поместите *Реверб* (см. *рис*. 41).

9.7. Сделайте также реверберацию для дорожек:

1. ***SD*** с готовыми настройками *Medium* (средняя комната).

2. ***Solo Guitar*** с готовыми настройками *Small* (маленькая комната).

Задание 10. Мастеринг

Итак, осталась заключительная часть — придать звучание нашей композиции соответствующе выбранному стилю и устранить общие оставшиеся дефекты. Делать это можно в этом же проекте, можно создать отдельный проект. Мы сделаем это в отдельном проекте.

Трек зашкаливает по громкости

Рис. 42

Первое, что нужно сделать, — это проверить, чтобы наш сведенный проект не зашкаливал по общей громкости и, как следствие этого, не было искажений в звуке. Откройте микшер и посмотрите дорожку общего звучания (мастер-шину).

Если горит красная кнопка *Clip*, значит, наш проект зашкаливает по громкости и ее нужно уменьшить. Делается это на этой же дорожке регулятором громкости. Сделайте так, чтобы общая громкость была в районе *–6 дБ*.

Сохраните файл, полученный после сведения.

10.1. Для этого нам нужно выделить область проекта, которую мы хотим сохранить.

Подведите курсор к верхней части поля сетки метронома, пока не появится значок в виде карандаша.

Ставим курсор для выделения:

Рис. 43

Растягиваем слева направо область выделения на столько тактов, на сколько нам нужно (до конца песни):

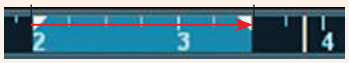


Рис. 44

Выделенная область должна быть окрашена в синий цвет. Область, окрашенная в красный цвет, означает неправильное выделение (справа налево).

Чтобы было удобнее растягивать, можно уменьшить масштаб ползунком внизу справа:

Рис. 45

10.2. После того как мы выделили сохраняемую область, можем сохранить сведенный файл.

Меню → File → Export → Audio Mixdown

Выбираем папку нашего проекта и сохраняем сведенный файл (см. *рис.* 46).

10.3. Теперь приступаем к мастерингу. Создайте новый проект, назовите его, например, *Mastering*. В нем создайте одну аудиодорожку с конфигурацией стерео. И через *Pool Window* импортируйте сведенный нами трек на эту дорожку.

10.4. Откройте миксер и поместите на сведенную дорожку эквалайзер. Увеличьте звук в районе *5 кГц*. Поэкспериментируйте с другими частотами.

10.5. Посмотрите на рисунок дорожки в проекте. Снова видим динамику уровня громкости. Сделаем звук по громкости ровнее с помощью компрессии. Поместите компрессор после эквалайзера (порядок, кстати, имеет значение, именно в таком порядке будет происходить обработка звука).

10.6. Настройте компрессор: Алгоритм — *Peak, Trashold — (–22) дБ, Ratio: 4:1, Attack — 0.1, Release — Auto, Make Up Gain — 5 дБ.*

10.7. Добавьте мультиполосную компрессию. Сделайте настройки по вкусу.

10.8. Отрегулируйте общую громкость проекта так, чтобы она не зашкаливала.

10.9. Сохраните полученный звуковой файл с расширениями *wave* и *mp3*.

Теперь можно закрыть *Nuendo* и насладиться своей работой, послушав файл на плеере. Сравните результат с исходной несведенной версией.

Заключение

Надеемся, что материалы этого практикума будут полезны для вас. Конечно, выполнив все задания, представленные в нем, вы не станете сразу же профессиональным звукорежиссером. Но предложенный практикум, по мнению авторов, может стать для вас первым шагом к достижению этой цели.

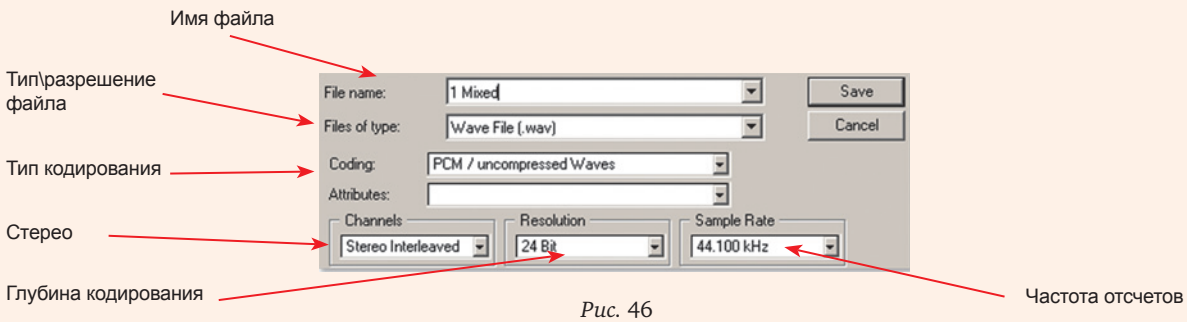
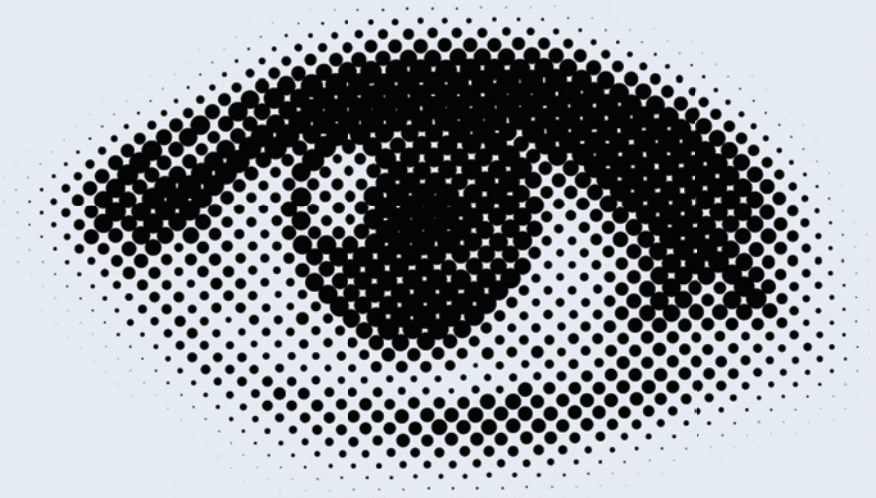


Рис. 46



Растровая графика

Графика на плоскости (2D)

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

Всем известно, что большую часть сведений и представлений об окружающем мире человек получает с помощью зрения. Далеко не все из того, что мы видим, можно без потери смысла описать в виде наборов цифр или текста. Поэтому очень важно существование средств представления информации “для зрения” и автоматических средств обработки такой информации.

Общее направление, в рамках которого решаются такие задачи, получило название компьютерной графики. Другими словами, ***компьютерной графикой*** *называют область деятельности, в которой компьютеры и программное обеспечение используются в качестве инструмента создания и обработки изображений.*

Из личного опыта вы прекрасно представляете себе, что такое “компьютерная графика”. По сути, это набор средств формирования изображения с помощью компьютера — всех средств, включая способы и аппаратуру создания, получения и хранения цифрового изображения, его отображения, передачи и обработки.

Разнообразие этих средств огромно, графика — одно из самых динамичных (и коммерчески востребованных) приложений всей ИТ-отрасли.

Первый вопрос, который нужно решить при организации обработки графических данных, — это представление и кодирование цвета.

В простейшем случае, когда возможных цветов всего два, используется один бит, состояние которого и задает цвет. Если же цветов становится больше, то такой подход уже не позволяет решить поставленную задачу.

Существует несколько способов кодирования цвета, применяемых при обработке графики.

Для описания градации одного цвета применяется обычное кодирование, в котором номер обозначает **градацию**. Чем больше значение, тем сильнее проявляется цвет. Для мониторов (в которых точка самостоятельно излучает свет) обычно 0 соответствует отсутствию цвета, а максимальное значение (например, 255) — максимальной светимости точки. Таким

образом, появляется возможность задавать **оттенок** на монохромном мониторе.

В случае, когда используется печатающее устройство и чернильная точка либо есть, либо нет, оттенок задается некоторой матрицей (например, 4×4 точки), количество цветовых точек которой формирует оттенок.

В более сложных случаях, когда речь идет о кодировании сложного цвета с большим количеством оттенков, рассматривают разложение цвета на несколько отдельных компонентов, которые, смешиваясь в одной точке, образуют заданный цвет.

Для каждого конкретного изображения все, что передается одним из компонентов цвета, также называется *каналом*. Компоненты цвета и способ образования из них видимого оттенка и представляют собой **цветовую модель**.

Цветовые модели разрабатывались в психологии восприятия задолго до появления вычислительной техники. Существует большое количество цветовых моделей, которые создавались и вводились разными авторами для описания и исследования зрения человека. С появлением проекционной и печатающей аппаратуры, с учетом технических требований были разработаны новые модели, учитывающие в первую очередь физические и технические аспекты формирования конкретного цвета.

Для формирования изображения на экране мы будем пользоваться **аддитивной цветовой моделью RGB** (*рис.* 1), в которой цвет образуется смешиванием трех компонентов:

- **R**ed — красного;
- **G**reen — зеленого;
- **B**lue — голубого.

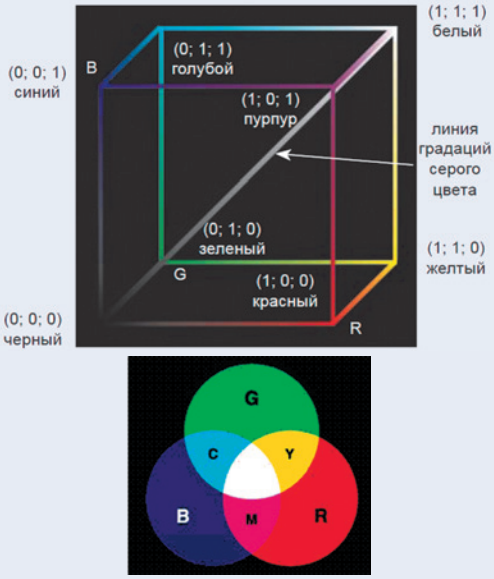


Рис. 1. Аддитивная цветовая модель RGB

Эта модель описывает цвет, который образуется из “суммы” света, излучаемого несколькими источниками. Такая модель является **аддитивной** (от лат. *additio* — прибавляю).

Самым популярным примером использования этой модели являются мониторы (в которых цвет каждого пикселя растра складывается из трех компонентов), проекторы и сканеры (которые чаще всего регистрируют отраженный свет).

Именно такая цветовая модель используется и в описании возможностей различных графических устройств. Цветовое пространство в этом случае описывают количеством битов, отводимых на сохранение цвета. Чаще всего используются режимы HighColor (16 бит, в соотношении 5:6:5 или 5:5:5) и TrueColor (24 бита, в соотношении 8:8:8).

Профессиональные программы обработки графической информации позволяют работать с расширенным представлением, когда на одну компоненту отводится не 8, а 16 бит.

Каждый компонент задается силой светимости, 0 соответствует отсутствию света. Таким образом, цвет 0-0-0 — это черный, цвет из равных долей каждого компонента — один из оттенков серого, а цвет с максимальными значениями компонентов — белый.

Второй вопрос, на который нужно ответить, создавая изображение с помощью компьютера, — это вопрос о том, что именно мы будем “расцвечивать” — а точнее, как мы будем графическую информацию хранить и показывать? От этого будут зависеть и обработка, и передача. Основных способов два:

Первый способ, которым чаще всего представляется графическая информация, получил название *растрового*. ***Растровая графика*** *— способ представления и хранения изображения в виде обозначения цвета точек (пикселей), находящихся в узлах прямоугольной равномерной координатной сетки —* ***растра***.

Если сетка достаточно плотная, то и точки получаются мелкие, поэтому человеческий глаз не воспринимает изображение как дискретное.

Основными параметрами изображения в растровой форме являются **разрешение линейное,** т.е. возможное количество точек на единицу площади, и **разрешение цветовое** — количество градаций цвета. Таким образом, различают разрешение линейное — количество столбцов по горизонтали и линий по вертикали, и цветовое/оттеночное — количество оттенков или цветов у каждой точки. Линейное разрешение описывают возможным количеством точек, а цветовое — в виде количества битов, отводимых на описание каждой отдельной точки.

Чем больше количество точек на единицу площади и количество цветов каждой точки, тем выше возможное качество изображения.

Второй способ — **векторная графика.** Растровая графика является чрезвычайно мощным способом представления и хранения изображений, особенно фотографических, но в ряде случаев прямое использование такого подхода неудобно. Это случаи, когда изображение создается из типовых элементов — **графических примитивов** (точек, прямых и кривых линий и т.п.). Представление таких элементов в виде точек лишает нас возможности менять параметры

примитива без перерисовки изображения, ограничивает возможности геометрических преобразований, требует много места при хранении.

Для преодоления этих ограничений применяется подход, подразумевающий хранение и обработку изображения не в виде растра, а в виде некоторых описаний отдельных элементов. Элементами обычно являются математические объекты с заданными конкретными параметрами. Параметры позволяют выполнить визуализацию элементов на устройстве вывода (**растеризацию**), исходя из его характеристик и заданного "окна" просмотра.

Поскольку пространственное положение примитивов и способ отображения задаются с помощью координат, этот способ хранения и обработки изображений получил название **векторной графики**.

Одним из наиболее существенных достоинств векторной формы представления изображения является ее компактность и малая зависимость объема от размеров изображения.

К минусам этой формы представления относится отсутствие общих стандартов (практически у каждого редактора есть свои собственные форматы и особенности) и высокие требования к системным ресурсам, особенно — вычислительным.

Тем не менее, если мы синтезируем изображение программно, то чаще всего именно векторную графику мы будем использовать как основу для работы.

Здесь мы не будем затрагивать вопрос обработки изображений с помощью специализированных графических пакетов, а попробуем описать основу их работы — основные приемы и алгоритмы, которые используют при их создании.

Для решения этих задач мы воспользуемся уже знакомой средой программирования — PascalABC.Net.

Обработка растровых изображений. Фильтры

Как мы знаем, множество растровых изображений мы получаем из реального мира — с помощью фотографий, видеосъемки, обработки значений датчиков и т.д. Полученное изображение, даже если оказывается достаточно качественным (что бывает далеко не всегда), часто требует изменений в зависимости от вашего замысла или предпочтений.

Нам требуются средства, которые позволяют менять растровое изображение целиком. Основой очень значительной части таких средств являются ***фильтры***.

Фильтры — это способ изменения изображения, при котором оно "пропускается" через преобразование и обрабатывается поточечно. Самый известный способ такого преобразования — применение *матрицы-свертки*, в которой цвет каждой точки — это сумма цветов окружающих ее точек с некоторыми коэффициентами.

К каждой точке растра, находящейся в зоне действия фильтра, применяется некоторое действие для расчета ее нового цвета. Очень часто это действие основано на *весовой матрице,* которая называется *ядром преобразования*, то есть матрице с нечетной длиной и шириной, в которой указан вес (то есть доля участия) пикселя в итоговом значении. Матрица накладывается на изображение так, чтобы ее центр (именно поэтому длина и ширина выбраны нечетными) совпал с изменяемым пикселем. Параметры пикселей изменяемой зоны, попавшие под матрицу, суммируются с весом, указанным в матрице, результат нормируется (т.е. пересчитывается в доли от общего веса) и записывается в центральный пиксель. После этого матрица сдвигается на один пиксель и все повторяется.

Вот простой фильтр ("размытие"):

1	1	1
1	1	1
1	1	1

При наложении на изображение действовать это будет примерно так:

14	4	55	44	44	55
12	64	44	221	33	34
22	55	22	127	22	22
12	44	33	67	44	55
11	33	77	23	23	22
135	44	65	34	22	34

Цвет в центральной точке будет рассчитан как сумма всех цветов внутри синего квадрата, деленный на суммарный вес матрицы-фильтра, с округлением до целого: (64 * 1 + 44 * 1 + 221 * * 1 + 55 * 1 + 22 * 1 + 127 * 1 + 44 * 1 + 33 * 1 + + 67 * 1)/9 ≈ 75.

Такой расчет делается для каждого компонента цвета.

Реализуем средство наложения такого фильтра и посмотрим, чего можно таким образом добиться.

Первое, что нам понадобится, — исходное изображение. Получение, сжатие, декодирование — сложные задачи, на решение которых здесь у нас нет времени. Воспользуемся готовыми средствами. В составе нашей среды есть основанные на средствах Windows библиотеки работы с изображениями.

```
uses ABCObjects,GraphABC;
var
// Специальный объект класса Picture
// для работы с изображением
  p : Picture;
begin
// Создание объекта из файла
  p := new Picture('example.jpg');
// Отрисовка изображения в рабочем окне
  p.Draw(0,0);
 while true do  ; // Ждем закрытия окна
end.
```

Для работы с изображением нам понадобятся средства для доступа к отдельным пикселям изображения. Для этого у объекта `Picture` есть мето-

ды GetPixel и setPixel — то есть получения и записи отдельного пикселя по его координатам.

Метод Picture.GetPixel вернет объект Color со свойствами R,G,B — которые нам и нужны. Изменить эти свойства нельзя, поэтому новое значение мы "соберем" из компонентов с помощью функции RGB(r,g,b).

Теперь мы реализуем функцию наложения фильтра. Она будет получать объект — исходное изображение, объект-результат для изображения с наложенным фильтром и сам фильтр.

Изначально, для простоты и понятности, фильтр будет целочисленной матрицей 5 × 5 (размер мы оговорим константой). Напомним — этот размер должен быть нечетным.

Мы будем накладывать эту матрицу целиком, начиная с точки (2,2) — чтобы она поместилась целиком. Перед наложением мы рассчитаем суммарный вес всех точек в матрице — чтобы выполнить нормирование. Результат применения — то есть цвет точки — будет называться ***откликом фильтра***.

```
const
 filterSize = 5;
 half = 2;
type
 filter = array[0..filterSize-1,
                0..filterSize-1] of real;
procedure filterApply(s,t: Picture;
                      filterMatrix: filter);
var
 i,j,ip,jp: integer;
 dv: real;
 rr,rg,rb: real;
 pnt: Color;
begin
// Рассчитаем делитель - для нормирования
 dv := 0;
 for i := 0 to filterSize-1 do
    for j := 0 to filterSize-1 do
       dv := dv + filterMatrix[i,j];
 if dv = 0 then dv := 1;
 for ip := half to s.Height-half-1 do
    for jp := half to s.Width-half-1 do
    begin
       rr := 0;
       rg := 0;
       rb := 0;
       // накладываем фильтр-свертку -
       // захватывая пиксели вокруг
       for i := -half to half do
          for j := -half to half do
          begin
            pnt := s.GetPixel(jp+j,ip+i);
            rr := rr+pnt.R*
            filterMatrix[i+half,j+half];
            rg := rg+pnt.G*
            filterMatrix[i+half,j+half];
            rb := rb+pnt.B*
            filterMatrix[i+half,j+half];
          end;
       t.SetPixel(jp,ip,RGB(round(rr/dv),
       round(rg/dv),round(rb/dv)));
    end;
end;
```

Рис. 2

Рис. 3

В основной программе ничего особенного происходить не будет:

```
begin
 p := new Picture('example.jpg');
 r := new Picture(p.Width ,p.Height);
 filterApply(p,r,blur);
 r.Draw(0,0);
 while true do
    ;
end
```

Имя фильтра blur мы пока не описали — это и есть матрица свертки. Содержание матрицы очень простое:

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

То есть в этом фильтре цвет по всем компонентам будет смесью окружающих точек. Опишем и применим его:

```
blur : filter := ( (1,1,1,1,1),(1,1,1,1,1),
    (1,1,1,1,1),(1,1,1,1,1),(1,1,1,1,1) );
```

Было — см. *рис.* 2. Стало — см. *рис.* 3.

Вот еще несколько фильтров на пробу:

Мягкое размытие:

0	1	2	1	0
1	3	10	3	1
1	10	90	10	1
1	3	10	3	1
0	1	2	1	0

Резкость:

0	0	0	0	0
0	–0.25	–0.25	–0.25	0
0	–0.25	3	–0.25	0
0	–0.25	–0.25	–0.25	0
0	0	0	0	0

Результат применения:

Рис. 4

У приведенных выше фильтров может быть несколько вариантов, которые вы можете без труда найти в сети и сравнить результаты. Отметим, что “сила”, с которой действует фильтр, будет зависеть не от величины коэффициентов, а от фактических размеров матрицы. Чем больше пикселей задействовано — тем заметнее эффект.

Теперь доработаем нашу функцию так, чтобы увеличить возможности применения фильтров. Для этого мы будем передавать (а не рассчитывать) коэффициент нормирования и величину смещения — для компенсации.

Функция будет выглядеть примерно так:

```
const
   filterSize = 5;
   half = 2;
type
   filter = array[0..filterSize-1,
                  0..filterSize-1] of real;
procedure filterApply(s,t : Picture;
filterMatrix : filter; dv,off : integer);
var
   i,j,ip,jp : integer;
   rr,rg,rb : real;
   pnt : Color;
begin
   for ip := half to s.Height-half-1 do
      for jp := half to s.Width-half-1 do
      begin
         rr := 0;
         rg := 0;
         rb := 0;
         for i := -half to half do
            for j := -half to half do
            begin
             pnt := s.GetPixel(jp+j,ip+i);
               rr := rr+pnt.R*
               filterMatrix[j+half,i+half];
               rg := rg+pnt.G*
               filterMatrix[j+half,i+half];
               rb := rb+pnt.B*
               filterMatrix[j+half,i+half];
            end;
         t.SetPixel(jp,ip,RGB
                  (test(round(rr/dv)+off),
         test(round(rg/dv)+off),
                  test(round(rb/dv)+off)));
      end;
end;
```

Обратите внимание: мы используем функцию `test`. Задача этой функции — убирать значения меньше 0 и больше 255. Здесь эта функция очень проста — на все значения меньше 0 возвращает 0, на все больше 255 — 255. Полагаем, читатели вполне в состоянии написать эту функцию сами.

Для повышения качества стоило бы не “обрубать” значения, а определить шкалу значений яркости и пересчитать все цвета пропорционально их положению на этой шкале.

С этими изменениями мы можем, например, повысить яркость изображения: просто добавив число ко всем компонентам цвета и применив нейтральный фильтр:

```
begin
   p := new Picture('example.jpg');
   r := new Picture(p.Width ,p.Height);
   filterApply(p,r,neutral,1,120);
   r.Draw(0,0);
   while true do;
end.
```

Еще один пример — инверсия цвета. Этот фильтр имеет только одно ненулевое значение — –1 в центре, а для “переворота” цвета добавляет смещение — 256.

Фильтров, приводящих к появлению самых разных эффектов, великое множество. Результаты зависят и от матрицы, и от того, как она накладывается, и от способов создания отклика и т.д.

Возможностей для экспериментирования у вас масса. Вот несколько вариантов:

1. Свертку накладывать на все пиксели, включая край. На край она накладывается частично — с пересчетом делителя нормирования.

2. Перед наложением край изображения масштабировать так, чтобы размеры стали больше на величину свертки.

3. Фильтр накладывать прямо на изображение — используются и уже обработанные, и еще не обработанные точки.

Задания для самостоятельной работы.

1. Размытие по Гауссиане может быть выполнено, в частности, с помощью такой свертки:

$$\begin{pmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 5 & 4 & 2 \\ 3 & 5 & 6 & 5 & 3 \\ 2 & 4 & 5 & 4 & 2 \\ 1 & 2 & 3 & 2 & 1 \end{pmatrix}$$

Примените его и сопоставьте результат с “обычным” размытием.

2. Один из практически важных фильтров — усреднение. В этом фильтре все значения в матрице сортируются, и отклик фильтра — среднее (медианное) значение. Реализуйте такой фильтр, проанализируйте его поведение для серых и цветных изображений.

3. Эффекта “Акварельности” изображения добиваются так: применяют фильтр усреднения, а потом к результату — фильтр резкости. Реализуйте этот комбинированный фильтр.

Графические примитивы. Растеризация

Следующая важная задача, которую решают при разработке средств машинной графики, — это отрисовка примитивов. Практически все современные аппаратные средства в машинной графике — растровые, то есть так или иначе опираются на координатную сетку пикселей. Но любой графический примитив — например, отрезок — состоит (а точнее — проходит) из нескольких пикселей. Возникает вопрос: какие из них нужно закрасить?

Приведем пример одного из основных алгоритмов, используемых в машинной графике в этом случае, — алгоритма построения отрезка. На его примере можно увидеть некоторые основные приемы решения таких задач.

Построение отрезка для нас означает заполнение тех точек растра, которые находятся на отрезке, соединяющем точки с заданными координатами. Очевидно, что просто вычислить эти координаты точно нельзя — поскольку не может быть “дробных” точек. Для простоты мы будем считать, что координаты определены именно в точках растра (т.е. их не нужно пересчитывать).

Самый распространенный алгоритм рисования отрезков получил название *алгоритма Брезенхема*.

Основная идея этого алгоритма — не использовать вычисления с плавающей точкой, а просто “пройти” по одной из осей и определить для каждой координаты в пределах линии, нужно ли для нее “сдвинуть” значение второй координаты.

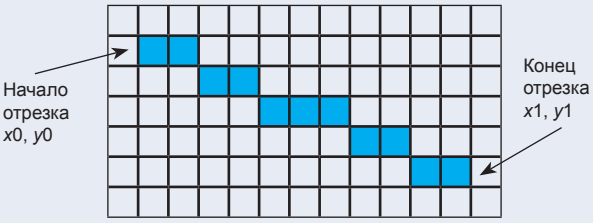


Рис. 5. Пример построения отрезка

Например, рассмотрим алгоритм для рисования линии на *рис.* 5, приняв координаты концов отрезка x_0, y_0 и x_1, y_1. Чтобы упростить понимание алгоритма, сначала запишем его на псевдокоде с использованием дробных чисел, считая, что исходное изображение — объект Image, все пиксели в нем — внутренняя матрица Pixel.

```
Deltax = x1 – x0
Deltay = y1 – y0
error = 0
deltaErr = Deltay/Deltax
y = y0
for x = x0 to x1
 Image.Pixel[x,y] = 1
 error = error + deltaErr
 if error > 0.5
      y = y + 1
      error = error – 1
```

Таким образом, мы вычислили коэффициент наклона, и увеличивали y тогда, когда прошли больше половины пикселя (то есть накопили ошибку в переменной `error` больше, чем 0,5). Чтобы перейти к целым числам, просто умножим дробные числа на Deltax и на 2 (там, где использовался коэффициент 0,5):

```
Deltax = x1 – x0
Deltay = y1 – y0
error = 0
deltaErr = Deltay
y = y0
for x = x0 to x1
 Image.Pixel[x,y] = 1
 error = error + deltaErr
 if 2*error > Deltax
      y = y + 1
      error = error – Deltax
```

Чтобы получить полный алгоритм, нужно учесть и другой случай наклона отрезка, когда y меняется быстрее, чем x. Обычно их учитывают, меняя местами координаты и направления сдвига точек. Аналогично можно реализовать алгоритмы построения окружностей и эллипсов с осями, параллельными осям координат.

Алгоритм можно использовать и для большего разнообразия линий, например, задав шаблон в виде битовой карты (рисовать точку или не рисовать), получать пунктирные линии, использовать не просто точку, а фигуру сложной формы (кисть) и т.д.

Используем этот алгоритм для решения такой задачи: написать процедуру построения линии,

нарисованной треугольниками или любыми фигурами, вписанными в размер 4 × 4 пикселей. Примерно так:

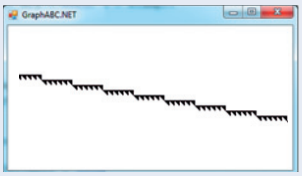


Рис. 6

Типовые встроенные функции такого сделать не позволяют: даже рисование произвольной кистью приведет к появлению сплошной линии.

Но мы знаем, как строится линия в принципе, — используем это для рисования линии не пикселями, а штампами 4 × 4.

Сначала напишем процедуру рисования штампом:

```
type
   stamp = array[0..3,0..3] of byte;
procedure DoStamp(s:Picture; x,y : word; c
: Color; brush : stamp);
var
   i,j : byte;
begin
   for i := 0 to 3 do
      for j := 0 to 3 do
         if brush[j,i] = 1 then
           s.SetPixel(x + j,y + i,c);
end;
```

Как видно, ничего сложного — штамп будет задан массивом 4 × 4, в котором там, где нужна точка цвета “c”, стоит 1.

Теперь рисование линии:

```
procedure MyLine(s: picture; x0,y0,x1,y1: Word;
                 c: Color; brush: stamp);
var
   Deltax, Deltay, error, DeltaErr: integer;
   x ,y, stepx, stepy : integer;
begin
   if (x1 < x0) then
      stepx := -4
   else
      stepx := 4;
   if (y1 < y0) then
      stepy := -4
   else
      stepy := 4;
   Deltax := abs(x1 - x0);
   Deltay := abs(y1 - y0);
   error  := 0;
   deltaErr := Deltay;
   y := y0;
   x := x0;
   while x <> x1 do
   begin
     DoStamp(s,x,y,c,brush);
     error := error + deltaErr;
     if 2*error > Deltax then
     begin
       y := y + stepy;
       error := error - Deltax;
     end;
     x := x + stepx;
   end;
end;
```

Единственное существенное отличие от алгоритма на псевдокоде — это учет того, что точки могут быть заданы в любом порядке, — это определит направление шага по каждой координате. Шаг равен 4 — поскольку это размер штампа.

Основная программа, как и в случае с фильтром, проста:

```
var
   p : Picture;
  triangle : stamp := ((1,0,0,0),(1,1,0,0),
                       (1,1,1,0),(1,1,1,1));
begin
  p := new Picture(500,500);
  MyLine(p,400,100,100,50,rgb(20,0,0),
         triangle);
  p.Draw(0,0);
  while true do;
end.
```

Как и в прошлом примере, простора для экспериментов много: размер и форма штампа, использование фона, режимы наложения и т.д.

Рисование отрезков — далеко не единственный алгоритм Брезенхема. Аналогично строятся, например, эллипсы и дуги эллипсов.

Основы векторной графики. Преобразования координат

Исторически первые задачи, которые решались в компьютерной (машинной) графике, — это задачи создания изображения (графика, чертежа и т.д.). При подготовке таких рисунков очень часто возникает задача преобразования координат для отображения, преобразования объектов — масштабирования, поворота и т.д.

Такие задачи решаются методами аналитической геометрии с помощью преобразования координат.

Одна из базовых задач, которую мы решим таким способом, — это пересчет координат из математической системы координат — в экранную. Как вы знаете, в математике 0 — в центре или слева снизу, оси направлены слева направо и снизу вверх. На экране точка отсчета сверху слева, и ось *y* направлена вниз. Придется пересчитать.

Предположим, что у нас есть область для отображения, заданная координатами углов (X_0, Y_0), (X_1, Y_1). Нам нужно отобразить в ней объект (например, график функции), координаты которого укладываются в пределы (x_0, y_0), (x_1, y_1).

Тогда предварительно рассчитаем коэффициенты: $kx = (X_1 - X_0)/(x_1 - x_0)$ и $ky = (Y_1 - Y_0)/(y_1 - y_0)$.

Координаты точки на экране можно будет рассчитать так:

$$\begin{cases} X = [(x - x_0) \cdot kx + X_0]; \\ Y = [(y - y_0) \cdot ky + Y_0]. \end{cases}$$

Квадратными скобками мы обозначим операцию получения целой части числа, хотя в программе значения следует округлять. Если важно соблюдать масштаб, то коэффициент выбирается один — наименьший.

Фигуру, заданную координатами точек (всех или только ключевых), можно легко преобразовывать.

Сдвиг точки — это просто добавление чисел (координат вектора) к координатам:

$$\begin{cases} x' = x + a; \\ y' = y + b. \end{cases}$$

Увеличить или уменьшить фигуру (и не обязательно оставлять ее на месте) можно просто умножив координаты ее точек на коэффициент. Если фигура должна сохранить логическое положение на "листе", то координаты предварительно придется пересчитать в систему координат с центром в середине фигуры (x_0,y_0), определяемым как среднее арифметическое координат:

$$\begin{cases} x' = x_0 + k * (x - x_0); \\ y' = y_0 + k * (y - y_0). \end{cases}$$

Наконец, поворот точки на угол вокруг начала координат будет описываться так:

$$\begin{cases} x' = x * \cos\alpha - y * \sin\alpha; \\ y' = x * \sin\alpha + y * \cos\alpha. \end{cases}$$

Если поворот нужен не вокруг начала — мы можем использовать перенос координат.

Важное свойство таких преобразований — сохранение формы фигуры. То есть линия преобразуется в линию, а поэтому можно пересчитать только координаты начала и конца отрезка, а промежуточные точки рассчитывать заново не надо.

Используя эту теорию, решим задачу синтеза достаточно сложного изображения: многолучевой звезды. Пусть для определенности лучей будет 7.

Начнем с постановки задачи. Звезда фактически представляет собой симметричный относительно центра многоугольник, в котором используются два радиуса: длинный и короткий. При этом "чередуются" они через равные углы. Вершины соединяются между собой.

Пусть первая вершина на оси ординат, тогда остальные вершины мы сможем вычислить поворотом на нужный угол.

Для реализации нам потребуется несколько вещей:

- во-первых, типы для хранения координат — математических и экранных;
- во-вторых, функции для выполнения поворота точки на угол;
- в-третьих, средства рисования ломаной линии по точкам.

Первая часть ничего сложного не представляет:

```
type
   pointR = record
      x,y : real;
   end;
   pointS = record
      x,y : word;
   end;
   screenCalc = record
      kx,ky : real;
      x0,y0 : word;
      xm0,ym0 : real;
   end;
```

Тут две простых структуры с координатами и структура для хранения данных пересчета (по формулам, которые мы приводили).

Теперь можно написать функции преобразований:

```
// Расчет коэффициентов преобразования
// в экранные координаты
function calcScreen(x0,y0,x1,y1 : word;
      xm0,ym0,xm1,ym1 : real) :screenCalc;
begin
   Result.kx  := (x1-x0)/(xm1-xm0);
   Result.ky  := (y1-y0)/(ym1-ym0);
   Result.x0  := x0;
   Result.y0  := y0;
   Result.xm0 := xm0;
   Result.ym0 := ym0;
end;
// Перевод математических координат
// в экранные
function toScreen(math : pointR;
         screen : screenCalc) : pointS;
var
   res : pointS;
begin
   res.x := round((math.x-screen.xm0)*
            screen.kx+screen.x0);
  res.y := round((math.y-screen.ym0)*
           screen.ky+screen.y0);
   toScreen := res;
end;
// Поворот точки на угол (в радианах)
function Rotate(math: pointR;
                angle: real) : PointR;
var
   res : pointR;
begin
   res.x := math.x*cos(angle)-math.y*
            sin(angle);
   res.y := math.x*sin(angle)+math.y*
            cos(angle);
   rotate := res;
end;
// Масштабирование относительно центра
// координат
function Scale(math : pointR; k : real) :
PointR;
var
   res : pointR;
begin
   res.x := math.x*k;
   res.y := math.y*k;
   Scale := res;
end;
```

Как видите, ничего сложного в этих функциях нет — все эти формулы вы видели либо тут, либо на уроках геометрии. Функция масштабирования нам понадобится чуть позже.

С хранением и отрисовкой линии чуть сложнее. В общем виде мы не знаем, сколько точек будет в ломаной. Эту задачу мы решим с помощью списка, добавив в раздел `type`:

```
   PList = ^PointList;
   PointList = record
     point : PointR;
     next  : PList;
   end;
```

И в функции:

```
function addPoint(coord : pointR;
                  next : PList) : PList;
begin
   new (Result);
   Result^.point := coord;
   Result^.next := next;
end;
```

И, наконец, функция отрисовки линии:

```
procedure Polyline(p : Picture; vertex :
PList; screen :screenCalc);
var
   test : PList;
   p1,p2 : pointS;
begin
  // Построение линии
    p1 := toScreen(vertex^.point,screen);
  //Первая точка списка
   test := vertex^.next; //Со второй точки
    // Пройдем по всему списку
    while test <> nil do
    begin
      p2 := p1;
      p1 := toScreen(test^.point,screen);
      p.Line(p2.x,p2.y,p1.x,p1.y);
      test := test^.next;
    end;
    p2 := toScreen(vertex^.point,screen);
    // Замкнем ломаную
    p.Line(p2.x,p2.y,p1.x,p1.y);
end;
```

Обратите внимание: мы перед собственно рисованием линии рассчитываем ее экранные координаты.

В секцию `Const` добавим определение количества точек-вершин:

```
const
  StarVertex = 14;
```

И теперь мы можем привести основную программу:

```
var
   star, test : PList;
   i : word;
   start1,start2 : pointR;
   screen : screenCalc;
   p : Picture;
begin
   start1.x := 10;
   start1.y := 0;
   start2.x := 5;
   start2.y := 0;
   star := addPoint(start1,nil);
   star := addPoint(Rotate(start2,2*
                   Pi/StarVertex),star);
  for i := 3 to StarVertex do
   if i mod 2 = 1 then
    star := addPoint(Rotate(start1,
           (2*Pi/StarVertex)*(i-1)),star)
   else
    star := addPoint(Rotate(start2,
          (2*Pi/StarVertex)*(i-1)),star);
 screen := calcScreen
             (0,0,500,500,-36,-36,36,36);
   p := new Picture( 500,500);
   PolyLine(p,star,screen);
   p.Draw(0,0);
   while true do;
end.
```

Результат исполнения:

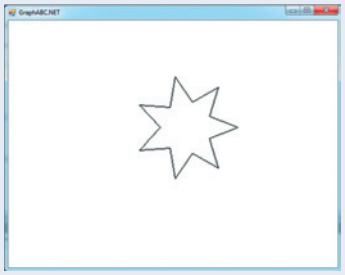


Рис. 7

Усложним задачу — построим 14 таких звезд, вложенных друг в друга. Решить такую задачу средствами только растровой графики трудно, а векторной — ничего сложного.

Изменим часть, посвященную собственно рисованию линии, — выполним после каждой отрисовки ее масштабирование:

```
for i := 1 to 14 do
   begin
    PolyLine(p,star,screen);
    // Пройдем по всему списку
    test := star;
    while test <> nil do
    begin
    test^.point := Scale(test^.point,1.1);
    test := test^.next;
    end;
   end;
```

Результат исполнения:

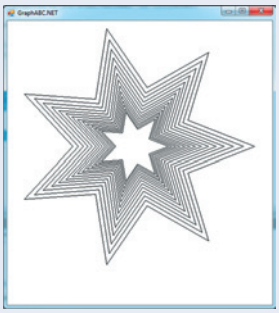


Рис. 8

Как видите, используя этот аппарат, мы можем строить сложные изображения, опираясь на простые средства, без потери качества изображения выполняя целый ряд преобразований. Стоит заметить, что возможности эти будут напрямую зависеть от вашей математической подготовки.

Задания для самостоятельной работы

1. Нарисуйте с помощью описанных средств фигуры-стрелки по кругу, диаметром 10 математических единиц.

2. Реализуйте набор звезд, в котором будет плавно меняться цвет: каждая звезда рисуется одним своим цветом.

3. Реализуйте картинку из 10 вложенных квадратов. Вершины внутреннего квадрата — середины сторон наружного.

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

Введение в SQL: практикум

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

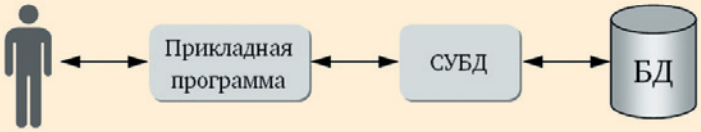
Введение

Современные информационные системы, использующие базы данных (БД) [1], чаще всего строятся по принципу двухступенчатого управления: пользователь работает с прикладной программой, которая передает команды системе управления базой данных (СУБД).

Прикладная программа обеспечивает дружественный интерфейс: большинство действий выполняется с помощью мыши, и дальнейший ход выполнения операций скрыт от пользователя. На базовом уровне изучения информатики этим вполне можно и ограничиться. Однако для тех, кто осваивает предмет на углубленном уровне и готовится к профессиональному освоению информационных технологий, важно представлять, как работает вся цепочка управления данными. На взгляд автора, именно понимание механизмов работы всей системы и ее составляющих — это тот ключ, который позволяет специалисту грамотно решать нестандартные задачи, или, говоря "школьным" языком, те задачи, "которые мы раньше не решали".

Большинство распространенных СУБД для управления данными использует язык SQL [2] (англ. — ***S**tructured **Q**uery **L**anguage*, язык структурных запросов). Задача прикладной программы — на основе команд пользователя построить соответствующий SQL-запрос, передать его СУБД, получить ответ и вывести его на экран.

Нередко специалисты по базам данных выступают за то, чтобы изучать базы данных в старшей школе именно на основе SQL, установив, например, СУБД на сервере в локальной сети. Такой подход не приобрел заметной популярности главным образом из-за того, что школьники, изучив тему "Базы данных" в



основной школе, привыкли к “мышиному” управлению и визуальному построению запросов, и поэтому воспринимают резкий переход к работе с текстовыми командами SQL как “ненужные” сложности.

В настоящей статье предлагается несколько иной подход, предусматривающий как использование визуального управления, так и параллельное изучение SQL-запросов. Это становится возможным в таких средах, как *OpenOffice Base* (*LibreOffice Base*) или *Microsoft Access*, которые, с точки зрения пользователя, объединяют в себе возможности СУБД и прикладной программы. В этих программах можно составлять запросы как в привычном для пользователей табличном виде (в формате QBE, ***Q**uery **b**y **E**xample,* запрос по образцу), так и на языке SQL. Сначала школьники осваивают визуальное построение запросов, время от времени “заглядывая” в режим SQL, а затем переходят к полному управлению БД с помощью SQL.

Далее представлен практикум “Введение в SQL” для учащихся, изучающих углубленный курс информатики по учебнику [3] на базе свободного программного продукта *OpenOffice Base* (или *LibreOffice Base*). Он состоит из двух практических работ — в первой используется однотабличная база данных, во второй — база данных с тремя связанными таблицами. Особенности выполнения работ с помощью *Microsoft Access* отмечены в комментариях внизу страницы.

Предполагается, что учащиеся уже выполнили практические работы по составлению запросов для однотабличных и многотабличных баз данных в конструкторе запросов и в ходе их выполнения познакомились с тем, как выглядят запросы на выборку данных на языке SQL.

Работа 1. Однотабличная база данных

В этой работе вы познакомитесь с основными командами языка *SQL*:

CREATE TABLE	создать таблицу
SELECT	выбрать данные
UPDATE	изменить данные
DELETE	удалить данные
DROP	удалить таблицу

Цель работы — понять общий подход к управлению данными с помощью текстовых запросов. В описании работы приведены вопросы-задания, которые нужно выполнить самостоятельно. При проведении занятий в классе (или в форме домашней работы) текст задания может быть выдан в виде электронного документа, в котором для ответов на вопросы оставлены специальные поля.

Создание и заполнение таблиц

1. Создайте новую пустую базу данных *SQLbase*.

2. Выберите пункт верхнего меню *Сервис — SQL*[1] и введите следующую команду для создания таблицы[2]:

```
CREATE TABLE "Туры" (
    "Код" INTEGER NOT NULL PRIMARY KEY,
    "Страна" VARCHAR(50) NOT NULL,
    "Транспорт" VARCHAR(20) NOT NULL,
    "Цена" DECIMAL(20,2) NOT NULL )
```

В этой команде требуется создать таблицу (**CREATE TABLE**) с именем “Туры”. В таблице должно быть четыре поля:

Код — целое число (**INTEGER**), непустое (**NOT NULL**), первичный ключ таблицы (**PRIMARY KEY**);

Страна — строка (**VARCHAR**) длиной до 50 символов, непустое;

Транспорт — строка длиной до 20 символов, непустое;

Цена — поле для хранения денежной суммы[3] в виде числа с 20 значащими цифрами, из них две цифры в дробной части (**DECIMAL(20,2)**).

Названия таблиц и полей в OpenOffice заключаются в двойные кавычки!

3. Выполните эту команду (кнопка *Выполнить*). Для того чтобы увидеть созданную таблицу, нужно закрыть базу данных и открыть ее заново[4]. Проверьте, что таблица действительно создана.

4. Зайдите в режим *Конструктора* и проверьте свойства полей таблицы, которые мы задали с помощью команды **CREATE TABLE**.

5. Зайдите в окно *SQL* и выполните команду для добавления в базу новой записи:

```
INSERT INTO "Туры"
    VALUES(1, 'Финляндия', 'автобус', 1200)
```

Эта команда вставляет (**INSERT**) в таблицу “*Туры*” (**INTO "Туры"**) одну запись. После ключевого слова **VALUES** в скобках перечислены через запятую значения полей новой записи в том порядке, в котором они задавались при создании таблицы.

Символьные строки в значениях полей заключаются в апострофы!

6. Выполните еще одно добавление записи:

```
INSERT INTO "Туры"
  VALUES (1, 'Норвегия', 'самолет', 15000)
```

Какая ошибка произошла? В чем ее причина?

Как нужно изменить SQL-запрос, чтобы исправить эту ошибку?

7. Добавьте в таблицу с помощью SQL-запросов еще несколько записей:

Страна	Транспорт	Цена
Швеция	паром	9000 р.
Германия	автобус	15 700 р.
Греция	самолет	23 000 р.
Норвегия	автобус	8000 р.
Германия	самолет	19 000 р.

Выбор данных и сортировка

1. Создайте новый запрос в режиме SQL[5]

[1] В *MS Access* нужно создать новый запрос в конструкторе запросов и переключиться в режим SQL.

[2] В *MS Access* названия таблиц и полей заключаются в квадратные скобки. Если они состоят из одного слова, скобки можно не ставить.

[3] В *MS Access* для хранения денежных сумм можно использовать тип данных **MONEY** (деньги).

[4] *MS Access* сразу обновляет список таблиц, открывать базу заново не нужно.

[5] В *LibreOffice* можно не создавать отдельный запрос, а использовать окно *Сервис — SQL*, отметив флажок “Показать вывод операторов SELECT”.

```
SELECT * FROM "Туры"
```

и выполните его. Посмотрите на результат.

Этот оператор выберет (**SELECT**) все поля (*) всех записей из таблицы "*Туры*" (**FROM "Туры"**).

2. Вместо "*" можно указать через запятую список нужных полей:

```
SELECT "Страна", "Цена" FROM "Туры"
```

Проверьте результат выполнения этого запроса.

3. Чаще всего нужно выбрать только записи, удовлетворяющие некоторому условию. Для этого используется ключевое слово **WHERE**, после которого стоит условие:

```
SELECT * FROM "Туры"
WHERE "Страна" = 'Норвегия'
```

Проверьте работу этого оператора.

Составьте запрос, который выбирает из таблицы "Туры" значения полей "Страна", "Транспорт" и "Цена" для всех автобусных туров.

Составьте запрос, который выбирает из таблицы "Туры" значения всех полей для туров с ценой меньше 10 000 руб.

4. Для того чтобы отсортировать данные по некоторому полю, в запросе после ключевых слов **ORDER BY** (англ. — "упорядочить по") указывают название этого поля:

```
SELECT * FROM "Туры" ORDER BY "Цена"
```

Проверьте работу этого запроса.

Если в конце предыдущего запроса добавить через пробел слово **DESC** (англ. — *descending* — "нисходящий"), сортировка выполняется в обратном порядке.

Составьте запрос, который выбирает из таблицы "Туры" значения всех полей для туров с ценой больше 10 000 руб. и сортирует результаты по убыванию цены.

5. В запросах можно использовать стандартные функции: **MIN** — минимальное значение, **MAX** — максимальное значение, **AVG** — среднее значение, **SUM** — сумма, **COUNT** — количество записей. Например, следующий запрос определяет минимальное значение поля *Цена* среди всех записей:

```
SELECT MIN("Цена") FROM "Туры"
```

Результат этого запроса — одно число.

Составьте запрос, который находит среднюю цену для туров в Норвегию.

6. Результаты запросов можно использовать в других запросах — получается вложенный запрос. Например, запрос

```
SELECT * FROM "Туры" WHERE "Цена" =
    (SELECT MIN("Цена") FROM "Туры")
```

вернет данные о самом дешевом туре.

Составьте запрос, который находит тур минимальной цены на самолете.

Изменение и удаление данных

7. Для изменения записей используется оператор **UPDATE**. Запрос, приведенный ниже, увеличивает цены всех туров на 10%:

```
UPDATE "Туры" SET "Цена" = "Цена"*1.1
```

Проверьте, что данные в таблице "*Туры*" действительно изменились.

Авиакомпании в данный момент представляют скидку на билеты, так что цены всех туров на самолетах составляют 80% от исходных. Составьте и выполните соответствующий запрос. Какая стоимость получилась у тура в Грецию?

8. Скопируйте таблицу "*Туры*", назвав копию "*Туры2*". Удалите из новой таблицы все туры в Германию с помощью запроса

```
DELETE FROM "Туры2"
WHERE "Страна" = 'Германия'
```

Проверьте, что данные в таблице "*Туры2*" действительно изменились.

9. Удалите таблицу "*Туры2*", которая больше не нужна, с помощью запроса

```
DROP TABLE "Туры2"
```

Загрузите базу данных заново и убедитесь, что таблица "*Туры2*" действительно удалена.

10. Оператор **SCRIPT** позволяет сохранить базу данных в текстовом формате для ее переноса в другую СУБД[6]. Сохраните базу с помощью запроса

```
SCRIPT 'имя файла'
```

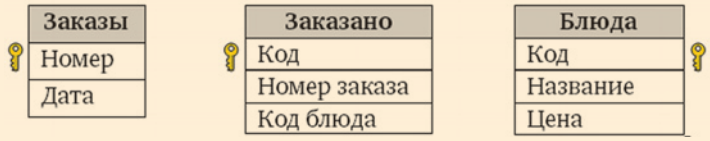
Откройте полученный файл в текстовом редакторе и изучите его.

Работа 2. Многотабличная база данных

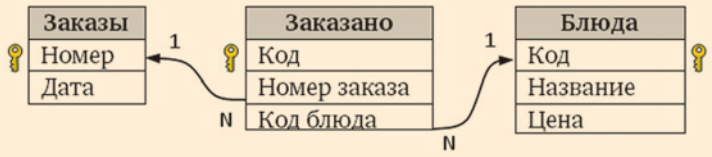
В этой работе вы познакомитесь с новой командой **ALTER TABLE** (изменить таблицу) языка *SQL* и научитесь составлять запросы к многотабличной реляционной базе данных.

Создание и заполнение таблиц

1. Используя только SQL-запросы, постройте три таблицы для реляционной базы данных кафе (значок 🔑 обозначает первичный ключ таблицы):



2. Теперь нужно добавить связи между таблицами:



Для этой цели используется команда **ALTER TABLE** (англ. — "изменить таблицу"). Построить связь между таблицами — это значит задать ограничение (**CONSTRAINT**), которое связывает первичный ключ одной таблицы с полем соответствующего типа другой. Если связываемое поле второй таблицы — неключевое, то оно называется *внешним*

[6] В *MS Access* сохранение базы в текстовом формате не поддерживается. Для решения этой задачи можно использовать программы сторонних фирм, например, *Export Table to SQL* (*http://www.ombelt.com/exporttable/sql/access/*).

ключом (**FOREIGN KEY**). Например, команда для создания связи “один ко многим” между ключевым полем *Номер* таблицы *Заказы* и неключевым полем *Номер заказа* таблицы *Заказано* выглядит так[7]:

```
ALTER TABLE "Заказано"
  ADD CONSTRAINT ORDER_NO
  FOREIGN KEY("Номер заказа")
  REFERENCES "Заказы"("Номер")
```

Такая запись дословно означает:

```
Изменить таблицу "Заказано"
добавить связь ORDER_NO
внешний ключ "Номер заказа"
ссылается на поле "Номер" таблицы "Заказы"
```

Здесь **ORDER_NO** — просто имя, которое мы выбрали для этой связи (можно было выбрать и другое).

3. Введите и выполните показанный выше SQL-запрос на добавление связи. Зайдите в меню *Сервис — Связи* и убедитесь, что связь действительно создана[8].

Самостоятельно составьте и выполните SQL-запрос на добавление второй связи.

4. С помощью SQL-запросов заполните базу следующими данными

Заказы

Номер	Дата
1	11.04.13
2	12.04.13

Заказано

Код	Номер заказа	Код блюда
1	1	1
2	1	3
3	1	4
4	2	1
5	2	2
6	2	2
7	2	5

Блюда

Код	Название	Цена
1	борщ	80 р.
2	бифштекс	110 р.
3	гуляш	70 р.
4	чай	10 р.
5	кофе	50 р.

Выбор данных и сортировка

5. Построим в режиме SQL запрос *СоставЗаказа*, который выводит номер заказа и название заказанных блюд. Эти данные находятся в двух таблицах — *Заказано* и *Блюда*, поэтому их нужно как-то объединить. Для этого используется связь “один ко многим” между таблицами, которую мы недавно установили. Действительно, для каждой записи в таблице *Заказано* нужно выбрать название блюда из таблицы *Блюда*, код которого совпадает с полем *Код блюда* таблицы *Заказано*. Это запрос на выборку данных, поэтому используем оператор **SELECT**:

```
SELECT "Заказано"."Номер заказа",
       "Блюда"."Название"
FROM "Заказано", "Блюда"
WHERE "Заказано"."Код блюда" = "Блюда"."Код"
```

Здесь из таблиц *Заказано* и *Блюда* выбираются поля *Номер заказа* и *Название*; условие в последней строке показывает, как объединять таблицы.

Поскольку названия полей в таблицах, из которых идет выбор, не совпадают, можно было записать запрос в сокращенной форме, указав после оператора **SELECT** только названия нужных полей:

```
SELECT "Номер заказа", "Название"
FROM "Заказано", "Блюда"
WHERE "Заказано"."Код блюда"="Блюда"."Код"
```

6. Теперь добавим в запрос дату заказа. Она находится в таблице *Заказы*, которая пока в запросе не участвует. Таким образом, нам нужно объединить три таблицы. Условие отбора получается сложным, два условия (связи по коду блюда между таблицами *Заказано* и *Блюда* и по номеру заказа между таблицами *Заказы* и *Заказано*) объединяются с помощью логической операции **AND** (И):

```
SELECT "Номер заказа", "Дата", "Название"
FROM "Заказано", "Блюда", "Заказы"
WHERE "Заказано"."Код блюда" = "Блюда"."Код"
AND "Заказано"."Номер заказа" =
    "Заказы"."Номер"
```

Проверьте результат выполнения этого запроса.

Измените запрос так, чтобы он выбирал только блюда из состава заказа № 1.

7. Построим еще один запрос *Итоги,* в котором для каждого заказа выводятся его номер, дата и общая сумма (с помощью функции **SUM**).

```
SELECT "Номер заказа", "Дата", SUM("Цена")
FROM "Заказано", "Блюда", "Заказы"
WHERE "Заказано"."Код блюда" = "Блюда"."Код"
AND "Заказано"."Номер заказа" =
    "Заказы"."Номер"
GROUP BY "Номер заказа", "Дата"
```

В последней строке указано, что по полям *Номер заказа* и *Дата* выполняется группировка, то есть сумма цен считается для каждой уникальной пары “*Номер заказа — Дата*”.

В таблице с результатами запроса заголовок столбца с суммой выглядит не совсем ясно для пользователя[9]:

```
'SUM("Блюда"."Цена")'
```

Для того чтобы сделать у этого столбца понятный заголовок “*Сумма*”, нужно добавить в первую строку запроса после **SUM("Цена")** так называемый “псевдоним” (подпись) с ключевым словом **AS**:

```
SELECT "Номер заказа", "Дата",
       SUM("Цена") AS "Сумма"
  ...
```

[7] В *MS Access* кавычки в названиях таблиц и полей нужно убрать, а название поля *Номер заказа* заключить в квадратные скобки, так как оно состоит из двух слов.

[8] В *OpenOffice Base* иногда связи появляются в окне *Связи* не сразу, а только после перезагрузки базы.

[9] В *MS Access* столбец будет называться примерно так: “*Expr1002*”.

Проверьте результат выполнения этого запроса. Псевдонимы можно задавать для всех значений, которые выводятся в запросе.

Измените запрос так, чтобы заказы были отсортированы в порядке убывания суммы (*используйте ключевые слова* **ORDER BY**).

Вложенные запросы

8. Построим запрос *МинСумма*, который выводит минимальную сумму заказа. Для этого будем использовать уже готовый запрос *Итоги*. Таким образом, источником данных для запроса *МинСумма* будет не таблица, а другой запрос. Отметим, что предварительно в запросе *Итоги* нужно отменить сортировку. Запрос получается очень простым

```
SELECT MIN("Сумма") AS "Сумма" FROM "Итоги"
```

9. Наконец, можно вывести информацию о заказе с минимальной суммой:

```
SELECT "Номер заказа", "Дата",
       "Итоги"."Сумма"
FROM "Итоги", "МинСумма"
WHERE "Итоги"."Сумма" = "МинСумма"."Сумма"
```

Обратите внимание, что этот запрос использует результаты выполнения двух ранее построенных запросов — *Итоги* и *МинСумма*. Запрос *МинСумма* можно было и не составлять, а вместо этого использовать вложенный запрос (запрос в запросе):

```
SELECT "Номер заказа", "Дата", "Сумма"
FROM "Итоги"
WHERE "Итоги"."Сумма" =
     (SELECT MIN("Сумма") FROM "Итоги")
```

Заметим, что если в базе данных есть данные о нескольких заказах с такой же (минимальной) суммой, будет показана информация обо всех этих заказах.

Измените запрос так, чтобы получить список всех заказов, сумма которых больше средней.

Для дальнейшего изучения языка SQL можно рекомендовать классическую книгу [2] и многочисленные электронные ресурсы в Интернете, например, [4]. Синтаксис языка SQL, который используется в *OpenOffice* и *LibreOffice*, изложен в официальной документации [5].

Выводы

Знакомство с языком SQL при изучении информатики на углубленном уровне помогает решить две задачи: 1) улучшить понимание работы реляционной СУБД не на уровне “щелкания мышкой”, а на уровне происходящего при этом обмена данными между компонентами системы; 2) подготовить учащихся к профессиональной работе с базами данных.

Использование сред *OpenOffice Base* (*LibreOffice Base*) и *MS Access,* объединяющих в себе функции прикладной программы и СУБД, позволяет достаточно легко перейти от визуального составления запросов к использованию языка *SQL*.

Литература

1. *Дейт К. Дж.* Введение в системы баз данных. М.: Вильямс, 2005.

2. *Грубер М.* Понимание SQL. М., 1993.

3. *Поляков К.Ю., Еремин Е.А.* Информатика. Углубленный уровень: учебник для 11-го класса: в 2 ч. М.: Бином, 2013.

4. Уроки по SQL и базам данных. Электронный ресурс *http://www.site-do.ru/db/db.php*.

5. HyperSQL User Guide. SQL Syntax. Электронный ресурс *http://www.hsqldb.org/doc/guide/ch09.html*.

Нереляционные базы данных: практикум

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

Введение

Подавляющее большинство современных баз данных (БД) основано на реляционной модели данных [1], предложенной в 1970 году американцем Э.Ф. Коддом из компании IBM. Это достаточно строгая математическая теория, которая строится на понятиях “отношение”, “домен”, “атрибут”, “кортеж”. На практике реляционными БД часто называют базы данных, представляющие собой набор взаимосвязанных таблиц. Для управления данными в таких базах используется язык SQL (***S**tructured **Q**uery **L**anguage* = язык структурных запросов).

Несмотря на то что за многие годы реляционные БД хорошо себя зарекомендовали, они не универсальны, и в некоторых задачах их применение приводит к серьезным проблемам.

Во-первых, для того чтобы построить надежно работающую реляционную БД, необходимо представить исходные данные как набор взаимосвязанных таблиц. Это требует серьезных усилий, кроме того, данные становятся менее понятными для человека, потому что он мыслит не таблицами, а объектами, которые имеют определенный набор свойств.

Во-вторых, данные об одном объекте разбросаны по разным таблицам, поэтому при поиске их приходится “вытаскивать” с помощью сложных SQL-запросов, которые выполняются сравнительно медленно при больших объемах данных.

В-третьих, во всех реляционных базах данных структура данных четко определена, причем она задается разработчиком при создании базы, и изменить ее весьма непросто. Теперь представим себе, что нам нужно хранить документы, которые могут иметь разное количество свойств с разными названиями и типами данных. В этом случае использовать реляционные БД по крайней мере очень сложно, поскольку они для этой цели не предназначены.

В-четвертых, объемы данных, которые нужно обрабатывать, все время возрастают, сейчас базы данных поисковых систем могут достигать нескольких петабайтов. Например, *Google* обрабатывает более 20 Пб данных в день, *Facebook* хранит 1,5 Пб фотографий, *Twitter* генерирует 2 Пб данных в год. Поэтому один компьютер с этим справиться не в силах (система может получать сотни тысяч запросов в секунду). Возникает задача: распределить нагрузку на большое количество (иногда десятки тысяч) серверов, связанных между собой через Интернет. Если при этом применять реляционную базу данных, для выполнения даже простых запросов нужно обращаться ко многим серверам, это недопустимо замедляет поиск. Подобная проблема возникает при "облачных" вычислениях, когда данные пользователя хранятся на серверах в Интернете.

Таким образом, реляционные БД хороши, когда вся база находится на одном компьютере, но они плохо масштабируются. Это значит, что при увеличении объема данных и количества запросов не удается увеличивать мощность системы, просто добавляя новые сервера. Следовательно, реляционная модель организации данных плохо подходит для распределенных информационных систем.

Чтобы решить эти проблемы, было предложено использовать так называемые «базы данных "ключ-значение"» (англ. — "*key-value*" *database*), которые строятся несколько иначе. Поскольку в них не используется реляционная модель данных, их часто называют *нереляционными*. Весьма популярен также и термин *NoSQL*, обозначающий "не только SQL" (англ. — ***N**ot **o**nly **SQL***). Он подчеркивает, что новые подходы не отрицают реляционную модель и язык SQL вообще, а просто предлагают другой способ организации данных, который в ряде задач обладает решающими преимуществами.

Базы данных "ключ-значение"

Базы данных "ключ-значение" можно представить себе как огромную таблицу, в каждой ячейке которой могут храниться произвольные данные ("значения"), их структура никак не ограничена. Каждому значению присваивается некоторый код ("ключ"), по которому его можно найти. СУБД поддерживается только добавление записи, поиск значения по ключу, а также изменение и удаление найденной таким образом записи. Никакие связи между значениями в явном виде не поддерживаются. Хотя объект может содержать ссылки на другие объекты (их ключи), СУБД не проверяет их правильность. Что касается надежности и целостности данных, эта задача возлагается не на СУБД, а на прикладную программу, которая работает с базой данных.

Все данные, относящиеся к конкретном объекту, хранятся в одном месте, поэтому при запросе не нужно обращаться к разным таблицам, а достаточно просто найти значение по ключу. Ключи объединяются в группы так, что все данные, связанные с ключами одной группы, хранятся на одном сервере. Таким образом, по ключу можно сразу определить нужный сервер и напрямую получить от него данные. За счет этого обеспечивается *масштабируемость* — если один сервер не справляется с нагрузкой, нужно добавить еще один и разделить данную группу ключей на две части.

В первую очередь базы данных "ключ-значение" используются при "облачных" вычислениях: в поисковой системе *Google* (система хранения данных *BigTable*), интернет-магазине *Amazon* (*www.amazon.com*, база данных *SimpleDB*), социальной сети *Facebook* (*www.facebook.com*, СУБД *Cassandra*), сервисе микроблогов *Twitter* (*twitter.com*, СУБД *Cassandra*).

В последние годы очень популярны *документоориентированные* БД, которые хранят *документы* — объекты, имеющие произвольный набор полей-свойств, например,

```
{
  ключ: 123123978623476239476923 7,
  автор: "А.С. Пушкин",
  название: "Евгений Онегин"
}
```

Важно, что другие документы в той же базе могут иметь совершенно другой набор полей.

Среди документоориентированных СУБД наиболее известны кроссплатформенные системы, относящиеся к классу свободного программного обеспечения: *MongoDB* (*www.mongodb.org*) и *CouchDB* (*couchdb.apache.org*).

Цель этой статьи — представить ознакомительный практикум по СУБД *MongoDB* в рамках изучения углубленного курса информатики в 11-м классе по учебнику [2].

Хранение данных

В отличие от реляционных баз данных СУБД *MongoDB* хранит данные не в виде таблиц, а в виде *коллекций документов*. *Документ* — это объект, имеющий свойства, которые задаются в виде пар "имя – значение". Главное свойство документа — это его *идентификатор* (код), который всегда играет роль первичного ключа.

В одной коллекции могут быть совершенно разные документы с разным набором свойств, это в первую очередь и отличает документоориентированную БД от реляционной.

Для управления данными в *MongoDB* используется язык *JavaScript*. Информация об объектах (документах) записывается в фигурных скобках, например:

```
{ name: "Вася", age: 16 }
```

Этот объект имеет два свойства (поля) — свойство "`name`" со значением "Вася" и свойство "`age`" со значением 16. Такой текстовый формат записи

называется *JSON* (англ. — ***J**ava**S**cript **O**bject **N**otation*, запись объектов с помощью *JavaScript*[1]).

Свойства объекта могут быть списками значений (массивами). Элементы массивов перечисляются в квадратных скобках через запятую:

```
{ name: "Вася", age: 16,
  lang: ["C", "Pascal", "JavaScript"] }
```

Здесь свойство "lang" — это массив, в котором записаны названия языков программирования.

Свойства могут сами быть объектами со своими свойствами, например:

```
{
  name: "Вася",
  family: { mother: "Вера",
            father: "Петя" }
}
```

Свойство family — это объект, содержащий два внутренних свойства (поля): "mother" и "father".

Чтобы данные занимали меньше места, при записи в память и передаче по сети они переводятся в двоичный формат *BSON* (англ. — ***B**inary **J**ava**S**cript **O**bject **N**otation*, двоичная запись объектов с помощью *JavaScript*).

Начало работы

Далее предполагается, что на компьютере, где выполняется работа, установлена и запущена СУБД *MongoDB* и консольная оболочка (см. Приложение). Большинство команд, описанных далее, работают и в веб-консоли на сайте *www.mongodb.org*. Для того чтобы открыть веб-консоль, нужно щелкнуть по кнопке 

в верхней части главной страницы этого сайта.

Сначала определим, с какой базой данных мы работаем, выполнив команду db (от англ. *database* — база данных):

```
db
```

Ответ "test" говорит о том, что СУБД автоматически подключилась к базе данных test.

В ходе работы мы построим простую базу данных для блога в Интернете, поэтому переключимся на базу данных blog с помощью команды

```
use blog
```

Если этой базы данных раньше не было, она будет создана. Проверьте, что она действительно стала активной.

Для добавления объекта в коллекцию используется оператор insert (*вставить*):

```
db.posts.insert (документ)
```

Первая часть записи, "db", означает обращение к рабочей (текущей) базе данных; вторая, "posts", — название коллекции (если такой коллекции нет, она будет создана), в скобках нужно записать информацию об объекте (документе), который заносится в коллекцию. В нашем примере в базе blog будет одна коллекция posts (сообщения, записи, "посты"). Для каждого поста нужно задать дату и текст. Добавьте одну запись следующим образом[2]:

```
db.posts.insert (
  {date: new Date("04/23/2013"),
   text: "Привет!"} )
```

Дата (свойство date) строится с помощью функции Date; ей передается символьная запись даты в формате, принятом в США (месяц/день/год).

Проверьте, создана ли коллекция posts с помощью команды, которая показывает все коллекции текущей базы:

```
show collections
```

Теперь проверьте, добавлена ли первая запись, выполнив команду

```
db.posts.find()
```

которая ищет и показывает все документы, входящие в коллекцию posts. Для вывода в красивом формате можно дополнительно вызвать функцию pretty (англ. — *приятный*):

```
db.posts.find().pretty()
```

В этом случае, если список свойств документа не помещается в одну строку, он выводится в столбик, например, так:

```
{
    "_id": ObjectId("5176abbc06a6380da34
966a2"),
  "date":
        ISODate("2013-04-22T20:00:00Z"),
   "text": "Привет!"
}
```

СУБД автоматически добавила поле "_id" (идентификатор, код), которое представляет собой суррогатный первичный ключ и строится с помощью функции ObjectId. Дата преобразована в формат Международной организации по стандартизации (ISO).

Если нужно, значение ключа "_id" можно задать самостоятельно, какое мы хотим. Удалим все документы из коллекции командой remove:

```
db.posts.remove()
```

— и добавим четыре новых поста в базу, указав явно идентификаторы:

```
db.posts.insert ( {_id: 1, date: new
Date("04/23/2013"),
  text: "Привет!" } )
db.posts.insert ( {_id: 2, date: new
Date("04/24/2013"),
  text: "Это второй пост." } )
db.posts.insert ( {_id: 3, date: new
Date("04/25/2013"),
  text: "Это третий пост." } )
db.posts.insert ( {_id: 4, date: new
Date("04/26/2013"),
  text: "Это четвертый пост." } )
```

[1] Формат JSON широко используется и в других языках программирования.

[2] Если вы работаете в операционной системе *Windows*, не используйте русские буквы в значениях свойств. Это связано с тем, что в *MongoDB* для хранения символьных строк используется кодировка UTF-8, которая не очень хорошо поддерживается в консоли *Windows*.

Убедитесь, что документы действительно добавлены в коллекцию.

Отметим, что, вместо того чтобы добавлять документы в консоли, можно создать в рабочем (текущем) каталоге текстовый файл с расширением `.js` (*JavaScript*), включающий эти команды, и выполнить его с помощью команды `load`, например:

```
load('data.js')
```

Поиск и сортировка

Для поиска нужных документов используется уже знакомая нам функция `find`. В скобках можно задать критерий поиска — это объект (записанный в фигурных скобках), который содержит название поля и значение этого поля, которое мы ищем. Например, найдем документ с кодом (идентификатором, "`_id`"), равным 2:

```
db.posts.find( {_id: 2} )
```

В условиях можно использовать не только строгие равенства, как в предыдущем примере, но и неравенства. Неравенство — это тоже объект, у которого специальное название свойства, начинающееся знаком "$":

`$ne` — не равно,

`$lt` — меньше, `$lte` — меньше или равно,

`$gt` — больше, `$gte` — больше или равно.

Например, вот так можно найти записи с идентификатором больше 2:

```
db.posts.find( { _id: {$gt: 2} } )
```

А так найдем посты, написанные 24.04.2013 или позднее:

```
db.posts.find( { date: {$gte: new Date("04/24/2013")} })
```

Для поиска можно использовать сразу несколько условий. Если все условия нужно выполнить одновременно, их записывают как один объект с несколькими свойствами. Например, следующий запрос находит все документы, у которых свойство "`_id`" больше `2`, а дата создания — не раньше 26.04.2013:

```
db.posts.find( { _id: {$gt: 2},
date: {$gte: new Date("04/26/2013")} } )
```

Самое мощное средство поиска в *MongoDB* — это оператор `$where`, которому можно передать строку в кавычках, задающую условие поиска на *JavaScript*, например,

```
db.posts.find( { $where: "this._id > 2" } )
```

Здесь `this` — это объект (документ), который требуется проверить; через точку записывается название нужного поля.

Оператору `$where` можно передать любую функцию на *JavaScript*, которая возвращает логическое значение (истинное, если условие отбора выполняется). Например, запрос со сложным условием, показанный выше, можно было записать так:

```
db.posts.find( { $where: function()
    { return this._id > 2  &&
      this.date >= new Date("04/26/2013")
                }
            } )
```

Пара символов "`&&`" в языке *JavaScript* обозначает логическую операцию "И", а символы "`||`" — логическую операцию "ИЛИ". Проверка на равенство записывается как "==", а условие "не равно" — как "!=".

Составьте запрос для поиска всех документов, у которых свойство `_id` *равно 1 или дата создания равна 26.04.2013.*

Для сортировки применяется функция `sort`. Она сортирует те документы, которые предварительно найдены с помощью `find`. При вызове функции `sort` в скобках указывается порядок сортировки — объект (в фигурных скобках), содержащий название поля для сортировки; значение этого поля может быть 1 (сортировка по возрастанию) или "–1" (по убыванию).

Эта команда отсортирует документы по возрастанию даты (от старых к новым)

```
db.posts.find().sort( {date: 1} )
```

а эта — по убыванию:

```
db.posts.find().sort( {date: -1} )
```

Изменение

Для изменения документов применяется команда `update`. Допустим, мы хотим добавить к документу с идентификатором 1 новое логическое свойство `visible` (англ. — *видимый*) и присвоить ему значение `false` (ложь), которое означает, что этот пост пока скрыт и выводить его на веб-страницу не нужно.

Функции `update` передаются два объекта в фигурных скобках: условие, позволяющее найти требуемый документ, и свойства, которые у него нужно изменить. Попробуйте выполнить команду

```
db.posts.update( {_id: 1},
                 {visible: false} )
```

и посмотрите, что получилось в результате. Оказывается, СУБД заменила весь объект с идентификатором 1 целиком на новый, содержащий (кроме свойства "`_id`") только одно свойство — "`visible`".

Теперь придется восстановить исходный документ. Сделайте это самостоятельно.

Для того чтобы не заменять документ полностью, а изменить (или добавить) значение какого-то поля, нужно использовать специальный объект со свойством `$set` (установить):

```
db.posts.update( {_id: 1},
            {$set: {visible: false}} )
```

Примените эту команду и проверьте результат ее выполнения.

> Обратите внимание, что сейчас в коллекции `posts` находятся документы с разной структурой: один из них имеет свойство `visible`, а остальные — нет. При этом никаких изменений в структуру самой базы данных вносить не пришлось.

Найдите все документы, у которых свойство `visible` *равно* `false`. *Какой запрос нужно для этого выполнить?*

Теперь выполним множественное обновление: установим свойство `visible` равным `true` (истина) для всех остальных документов, для которых это свойство не установлено (или, что то же самое, равно специальному нулевому значению `null`):

```
db.posts.update( {visible: null},
                 {$set: {visible: true}},
                 {multi: true} )
```

Третий параметр — объект со свойством `multi` (англ. — *множественный*), равным `true` (истина), разрешает изменение нескольких документов сразу (если его не указать, будет изменен только один документ — тот, который найден первым).

Составьте запрос для поиска всех документов, у которых свойство `visible` *равно* `true`*, а дата создания — не позднее 25.04.2013.*

Теперь добавим комментарий к одному из постов (с идентификатором 2). Заметьте, что благодаря документоориентированной СУБД заранее планировать наличие комментариев не нужно — мы можем добавлять новые свойства к любому документу "на ходу"[3].

Комментариев может быть много, поэтому новое свойство `comments` будет массивом. Для добавления нового элемента в массив используется специальный объект со свойством-командой `$push` (англ. — *втолкнуть*):

```
db.posts.update( {_id: 2},
{$push: {comments: "Комментарий 1"} })
db.posts.update( {_id: 2},
{$push: {comments: "Комментарий 2"} })
db.posts.update( {_id: 2},
{$push: {comments: "Комментарий 3"} })
```

Когда выполняется первая из этих команд, у документа с идентификатором 2 еще нет свойства `comments` — оно будет создано автоматически.

Удаление

Как вы уже знаете, для удаления всех документов из коллекции используется команда `remove`*. Запишите команду, которая удаляет все документы из коллекции* `posts`*, но не выполняйте ее.*

С помощью команды `remove` можно удалять отдельные записи — условие для их поиска задается как параметр функции. Например:

```
db.posts.remove( {_id: 4} )
```

Выполните эту команду и проверьте ее выполнение.

Запишите команду для удаления всех документов, у которых свойство `visible` *установлено в* `false`*, и выполните ее.*

С помощью команды `drop` вся коллекция удаляется из базы. Например,

```
db.posts.drop()
```

Объясните, в чем отличие между командами `remove()` *и* `drop()`.

[3] В то же время нужно учитывать, что расширение существующих документов (увеличение их размера) в *MongoDB* замедляет работу системы, вызывая перераспределение памяти.

Проверьте, что коллекция `posts` *действительно была удалена. Какую команду нужно для этого использовать?*

На этом выполнение практической работы закончено.

Как выбрать структуру данных?[4]

На многих сайтах поддерживаются вложенные комментарии. Это значит, что пользователь может отвечать не на исходное сообщение, а на конкретный комментарий. При этом возникает вопрос: как правильно организовать данные в документоориентированной БД?

Например, добавляемый комментарий можно представить как объект, содержащий два или три свойства: номер "`no`", текст "`text`" и, в том случае, когда комментарий является ответом на другой комментарий, номер "родительского" комментария "`parent`":

```
db.posts.update( {_id: 2},
 {$push: {comments: {no: 1, text:
          "Комментарий 1"}} })
db.posts.update( {_id: 2},
 {$push: {comments: {no: 2, text:
          "Ответ на 1", parent: 1}} })
db.posts.update( {_id: 2},
 {$push: {comments: {no: 3, text:
        "Ответ на ответ", parent: 2}} })
```

> Обратите внимание, что СУБД никак не обеспечивает целостность ссылок, то есть правильность значений свойства "`parent`". Мы могли бы задать для всех комментариев одинаковые номера "`no`" или установить свойство "`parent`", равное несуществующему номеру комментария. Поддержку целостности в этом случае должна обеспечивать прикладная программа, работающая с базой.

В данном случае проблема с поддержкой целостности возникла из-за того, что мы попытались перенести принципы проектирования реляционных баз данных на нереляционные. Однако есть и другой способ решения этой задачи: добавлять комментарии прямо к родительскому комментарию, как новые свойства. Каждый комментарий — это объект, содержащий в простейшем случае только текст:

```
{ text: "Комментарий 1"}
```

Как только к нему получен первый "дочерний" комментарий, мы добавляем новое свойство-массив `comments`, так что объект выглядит так:

```
{ text: "Комментарий 1",
  comments: [ { text: "Ответ 1 на 1"} ] }
```

Новые комментарии просто добавляются в массив `comments`:

```
{ text: "Комментарий 1",
  comments: [
              { text: "Ответ 1 на 1"},
```

[4] Материал для любознательных. В основную работу не входит.

```
            { text: "Ответ 2 на 1"}
          ]
}
```

Построим такой объект с помощью консоли *MongoDB*. Сначала добавляем комментарий к посту с идентификатором 2:

```
db.posts.update( {_id:2},
   {$push: {comments:
   {text:"Комментарий 1"}}} )
```

Теперь добавляем к нему два комментария:

```
db.posts.update({_id:2},
   {$push: {"comments.0.comments":
   {text:"Ответ 1 на 1"}}})
db.posts.update({_id:2},
   {$push: {"comments.0.comments":
   {text:"Ответ 2 на 1"}}})
```

Разберемся в довольно странной записи `"comments.0.comments"`. Дело в том, что свойство `comments` — это массив, а нумерация элементов массива в *JavaScript* начинается с нуля. Запись `"comments.0"` обозначает нулевой (первый по счету) элемент массива `comments`. У этого элемента есть свойство-массив `comments` (в первый раз этот массив будет создан), с которым мы далее и работаем.

Можно далее наращивать вложенность комментариев по такой же схеме:

```
{ text: "Комментарий 1",
  comments: [
  { text: "Ответ 1 на 1",
    comments: [
     { text: "Ответ на ответ" }
    ]
  } ]
}
```

Для создания такой структуры нужно добавить новый элемент к массиву

```
"comments.0.comments.0.comments":
db.posts.update({_id:2},
   {$push: {"comments.0.comments.0.comments":
   {text:"Ответ на ответ"}}})
```

Использование JavaScript

Все документы с комментариями можно найти с помощью объекта со свойством `$exists` (существует):

```
var p = db.posts.find( {comments:
{$exists: true}} )
```

Результат этого запроса не выводится на экран, а записывается в переменную `p`. Затем можно определить его длину (число найденных документов) с помощью функции `length` (англ. — *длина*) и вывести на экран в цикле в формате *JSON* с помощью функции `printjson`:

```
for(i = 0; i < p.length(); i++)
  printjson( p[i] )
```

Можно работать и с отдельными свойствами, например, вывести на экран даты всех найденных постов с помощью функции `print`:

```
for(i = 0; i < p.length(); i++)
  print( p[i].date )
```

Рассмотрим более сложную задачу: вывести все посты с комментариями первого и второго уровней. Будем считать, что список нужных постов уже находится в переменной `p`. Это массив, и можно перебрать все его элементы в цикле:

```
  for(i = 0; i < p.length(); i++) {
    print(p[i].text)  // вывести сам пост
  ... // вывести комментарии к посту p[i]
  }
```

Чтобы вывести комментарии, сначала определим, есть ли они у поста `p[i]`, — в этом случае свойство `comments` не равно нулевому значению. Тогда проходим по массиву `p[i].comments` и выводим текст каждого комментария (здесь `j` — номер комментария первого уровня):

```
if ( p[i].comments )
  for(j = 0; j < p[i].comments.length; j++) {
    print(" ", p[i].comments[j].text)
  ... // вывести все комментарии второго уровня
  }
```

Обратите внимание, что массив `p[i].comments` — это в отличие от массива `p` обычный массив *JavaScript* (а не массив документов *MongoDB*), поэтому здесь `length` — не функция, а свойство массива, и скобки после этого слова ставить не нужно.

Аналогично в цикле выводим комментарии второго уровня:

```
if ( p[i].comments[j].comments )
  for(k = 0;
    k < p[i].comments[j].comments.length;
    k++)
    print(" ",
    p[i].comments[j].comments[k].text)
```

Если необходимо показать комментарии всех уровней, удобно оформить рекурсивную функцию с двумя параметрами: `post` — документ, у которого могут быть комментарии, то есть свойство `comments`, и `margin` — отступ для того, чтобы комментарии разных уровней выводились “лесенкой”.

```
function printComments(post, margin) {
  if ( post.comments )
   for(i = 0;i < post.comments.length;i++)
   {
     print(margin,post.comments[i].text)
     printComments(post.comments[i],
                   margin+" ")
   }
}
```

Эту функцию можно вызывать в цикле для каждого найденного поста:

```
  for(i = 0; i < p.length();i++) {
  // вывести сам пост
    print(p[i].text)
  // вывести все комментарии
    printComments ( p[i], " " )
  }
```

Выводы

В документоориентированных базах данных информация о документе (объекте) не “разбросана” по нескольким таблицам, как в реляционных БД, а хранится в одном месте. С точки зрения теории баз данных такая база данных *денормализована*. С одной стороны, это существенно ускоряет поиск данных по ключу, с другой стороны, может привести к избыточности (повторному хранению одних и тех же данных) и логическим ошибкам.

Нереляционная БД позволяет хранить в одной коллекции документы самой разной структуры, причем структуру каждого документа (состав и значения свойств объектов) можно легко менять независимо от других без каких-либо административных переделок самой базы.

Серьезное преимущество таких БД — возможность масштабирования для создания распределенных информационных систем, в том числе в Интернете. Например, если один сервер не справляется с нагрузкой, все множество ключей документов, которые он обрабатывает, можно разделить на несколько групп, и каждую из этих групп поручить отдельному серверу. В этом случае центральный сервер выполняет роль диспетчера: получив запрос, он переправляет его на тот сервер, который хранит документ с заданным ключом. Для пользователя этот процесс абсолютно "прозрачен".

Некоторые нереляционные СУБД, в том числе *MongoDB*, поддерживают технологию Map/Reduce, которая была разработана компанией *Google* для параллельной обработки данных, расположенных на разных серверах [5].

Главный недостаток нереляционных СУБД состоит в том, что они не обеспечивают целостность данных. Это значит, что связи между объектами (документами) в принципе могут быть организованы, но СУБД их никак не проверяет. Эта задача целиком возлагается на прикладную программу.

Нереляционные СУБД нельзя рассматривать как универсальное решение. Это инструмент, который может подходить, а может и не подходить для решения конкретной задачи. Они применяются там, где:

- нужно быстро обрабатывать большие объемы данных в распределенных информационных системах;
- необходимо горизонтальное масштабирование, то есть размещение данных на большом количестве серверов с возможностью наращивания;
- требуется хранить данные различной структуры, причем структура документов может изменяться в любой момент;
- целостность и непротиворечивость данных некритична, или контроль этих свойств может быть "поручен" прикладной программе.

Для дальнейшего знакомства с СУБД *MongoDB* полезна официальная документация [3] и русский перевод материала *K.Seguin "The Little MongoDB Book"* [4]. На сайтах *mongolab.com* и *mongohq.com* можно разместить в облачных сервисах свои базы данных, управляемые *MongoDB*, в том числе базу размером 512 Мб — бесплатно.

Литература

1. *Дейт К. Дж.* Введение в системы баз данных. М.: Вильямс, 2005.
2. *Поляков К.Ю., Еремин Е.А.* Информатика. Углубленный уровень: учебник для 11-го класса: в 2 ч. М.: Бином, 2013.
3. Официальная документация по *MongoDB*: *http://docs.mongodb.org/manual*.
4. *Сегуин К.* Маленькая книга по *MongoDB* ("*The Little MongoDB Book*"). Пер. с англ. *http://jsman.ru/mongo-book/*.
5. Map/Reduce. Электронный ресурс *http://ru.wikipedia.org/wiki/MapReduce*.

ПРИЛОЖЕНИЕ

Установка и запуск MongoDB в ОС Windows

1. Скачайте архив с программой со страницы *http://www.mongodb.org/downloads*.
2. Распакуйте архив в отдельный каталог в любом месте диска, например, в каталог `C:\MongoDB`. Внутри этого каталога должен появиться каталог `bin`, в котором находятся все файлы СУБД.
3. Создайте новый каталог, где будут храниться данные. Удобно создать этот каталог прямо внутри каталога `C:\MongoDB`, например, `C:\MongoDB\data`.
4. Создайте в каталоге `C:\MongoDB\bin` командный файл `start.bat` с командой запуска серверной части:

```
C:\MongoDB\bin\mongod.exe --dbpath C:\MongoDB\data
```

С помощью параметра `dbpath` указывается путь к каталогу с базами данных.

5. Создайте (например, на рабочем столе) ярлык на файл `C:\MongoDB\bin\start.bat` для запуска серверной части.
6. Создайте ярлык на файл `C:\MongoDB\bin\mongo.exe` для запуска консоли (оболочки).
7. Запустите с помощью ярлыков серверную часть, затем — консоль.

Установка и запуск MongoDB в ОС Ubuntu

1. В Терминале введите следующую команду для импорта публичного GPG-ключа:

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv 7F0CEB10
```

2. Создайте файл `/etc/apt/sources.list.d/10gen.list` и добавьте в него строку для обращения к репозиторию компании 10gen:

```
deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen
```

3. Обновите информацию о пакетах в репозиториях командой

```
sudo apt-get update
```

4. Установите последнюю версию пакета командой

```
sudo apt-get install mongodb-10gen
```

5. Запустите серверную часть `mongod` как службу командой

```
sudo service mongodb start
```

6. Запустите командную оболочку (консоль) командой

```
mongo
```

Установка *MongoDB* в других операционных системах описывается на веб-сайте проекта *http://docs.mongodb.org/manual/installation/*.

ТАРИФНЫЕ ПЛАНЫ НА ПОДПИСКУ

на 2013/14 учебный год

для образовательных учреждений, участвующих в общероссийском проекте «Школа цифрового века»

ШКОЛА ЦИФРОВОГО ВЕКА

Электронные версии всех журналов для всех работников образовательного учреждения (доставка в Личные кабинеты)

электронная версия

4000 руб. за год –

оргвзнос от образовательного учреждения за участие в проекте независимо от количества педагогических работников

подробности на digital.1september.ru

Образовательные учреждения, участвующие в проекте «Школа цифрового века», могут воспользоваться эксклюзивными льготными тарифами для обеспечения школьных библиотек бумажными версиями и CD-версиями предметно-методических журналов Издательского дома «Первое сентября»:

- тариф **«Школьная библиотека»** – комплект бумажных версий всех журналов Издательского дома «Первое сентября» (доставка по почте)
- тариф **«Школьная медиатека»** – комплект CD-дисков с электронными версиями журналов и методическими материалами (доставка по почте)

Подробности — в Личном кабинете школьного администратора проекта «Школа цифрового века» (раздел «Спецпредложения»)

ШКОЛЬНАЯ БИБЛИОТЕКА

Комплект бумажных версий журналов и CD с дополнительными материалами для практической работы (доставка по почте)

бумажная версия

19 000 руб. за полгода

для ОУ – участников проекта «Школа цифрового века»

(для сравнения: стоимость подписки на комплект на почте – 23 000 руб. за полгода)

ШКОЛЬНАЯ МЕДИАТЕКА

Комплект CD с электронными версиями журналов и дополнительными материалами для практической работы (доставка по почте)

CD-версия

9000 руб. за полгода

для ОУ – участников проекта «Школа цифрового века»

(для сравнения: стоимость подписки на комплект на почте – 13 000 руб. за полгода)

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

VISA MasterCard QIWI WebMoney

Информатика в облаках

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

Облачные вычисления, облачные хранилища данных, облачные платформы — все эти термины прочно закрепились в лексиконе современного программиста, разработчика десктоп- и онлайн-приложений. В 2012 г. “Информатика” уже публиковала обзор облачных хранилищ данных [1]. Здесь мы рассмотрим более широкий класс сетевых объектов — облачные решения — и приведем примеры того, как их можно применять при обучении информатике и ИКТ.

Как и в других подобных случаях, сама по себе идея, технологическая концепция появилась задолго до того, как с ней стал ассоциироваться модный термин. Так, например, асинхронная передача данных с помощью объекта XMLHttpRequest была внедрена в браузере Microsoft Internet Explorer 5.0 в марте 1999 г. (а с помощью тега IFrame тремя годами ранее), но революция в архитектуре веб-приложений произошла гораздо позже; аббревиатура AJAX впервые была использована только в 2005 г., а массовый переход на одностраничные AJAX-интерфейсы происходит в 2010–2013 гг.

В 2006 г. компания Amazon представила бета-релиз сервиса Elastic Compute Cloud, который позволял клиентам работать в виртуальной вычислительной среде (virtual computing environment) и оперировать в “новом мире масштабируемых веб-сервисов по запросу” [2]. Примерно через год слово “cloud” — “облако” — стало появляться в публикациях в составе упомянутых в начале этой статьи словосочетаний. Но сама концепция распределенных вычислений — виртуального суперкомпьютера, состоящего из мощностей, относящихся к разным сетям, — фактически использовалась гораздо раньше и получила название “Грид” (*Grid* — сеть, решетка) [3]. В 2004 г. Иэн Фостер и Карл Кессельман определили вычислительную сеть (вычислительный грид) как “инфраструктуру из программного и аппаратного обеспечения, которая предоставляет отказоустойчивый стабильный обширный и недорогой доступ к высокопроизводительным вычислительным мощностям”, сравнивая доступ к ней с доступом к электрической сети

(power grid). В 1999 г. популярность приобрел проект SETI@Home, основанный на идее добровольного присоединения компьютеров к виртуальной вычислительной сети с целью поиска радиосигналов внеземных цивилизаций. Спустя 10 лет компьютеры, заражаемые вирусами, принудительно становятся участниками бот-сетей, используемых для взлома защищенных ресурсов.

Термин "Грид" был довольно быстро замещен термином "облако"; суть функционирования таких систем не изменилась. Но представляется, что главное отличие известных нам сегодня облачных систем от грид-систем находится в контексте взаимоотношений Всемирной паутины (веб) и Интернета. Веб (WWW) является, с одной стороны, результатом работы одного из протоколов Интернета (HTTP), а с другой — универсальным интерфейсом доступа ко всем ресурсам Интернета (через веб-сайты пользователи проверяют электронную почту, просматривают FTP-архивы, получают доступ к потоковому видео, хотя эти виды данных передаются каждый по своему протоколу). Так и современные облачные решения приобрели свою огромную популярность благодаря ориентированности на доступ к ресурсам и управление ими через веб. То же можно сказать и об информатике как школьном предмете: содержание осваивается одновременно (параллельно, распределенно) на различных предметах и вне предметов (в рамках индивидуальных и групповых проектов, внеурочной деятельности вообще). Это обучение в электронной образовательной информационной среде (ЭОИС) [4] в соответствии с Законом об образовании [5] называется **электронным**; для него характерна перемещаемость рабочего места: рабочие материалы находятся не на локальном носителе информации (и не на физическом), а на сетевом ресурсе глобального (но защищенного) доступа. В этом случае оно является **мобильным**. Мобильность обучения указывается учеными в качестве одного из условий формирования Smart-образования и Smart-общества, поскольку гарантирует мгновенное включение любого формируемого в процессе получения образования контента в IT-системы [6,7].

Таким образом оказывается, что современное электронное (мобильное) обучение информатике и ИКТ подобно облачным вычислениям как концептуально, так и технически.

Предположительно, для изучения **практически любой** темы по информатике можно подобрать совокупность онлайновых (облачных) решений (технологий, инструментов), позволяющих изучать ее только с помощью браузера, не пользуясь другими десктоп-приложениями.

Преимущества приложений, запускаемых в браузере, довольно очевидны:

- не требуется установка приложения (и, как следствие, права администратора);
- обновление версий автоматическое;
- приложения кроссплатформенны: можно использовать приложение на любом компьютере, имеющем соединение с Интернетом;
- все данные хранятся в Интернете, что исключает зависимость от физических носителей;
- данные хранятся на нескольких серверах с многоуровневой защитой от атак и порчи;
- при работе веб-приложения компьютер пользователя гораздо меньше подвержен опасности вирусного заражения, чем при запуске exe-файлов;
- веб-приложения "чисты" в аспекте лицензий и авторского права.

К числу очевидных недостатков относится зависимость от стабильного интернет-соединения и от вендора (поставщика приложения), который волен как угодно менять интерфейс и функциональность облачного ресурса. Например, на сайте *aviary.com* длительное время работал набор онлайн-редакторов изображений, аудио и видео, доступ к которому был впоследствии прекращен владельцами сервера.

Освоение облачных приложений — это деятельность, которая требует от учащихся анализировать меняющуюся ситуацию, искать знакомые элементы в новых интерфейсах, адаптироваться, подбирая инструменты для возникающих задач. Десктоп-приложения отличаются гораздо большей стабильностью, и, установив, например, офисный пакет, пользователь может работать с одним и тем же интерфейсом годами. Во всяком случае, это может продолжаться до тех пор, пока вендор не прекратит поддержку данной версии и не вынудит пользователя мигрировать к новой версии. Например, в 2007 г. Microsoft принудительно перевел пользователей Microsoft Office на Ribbon-интерфейс, что заставило множество потребителей заново осваивать принципы работы с пакетом, но это скорее исключительная ситуация.

В 2010 г. Microsoft объявил о создании решения Office 365, призванного объединить в облачном сервисе все онлайн-продукты, включая Office Web Apps — веб-базированную версию Word, Excel, PowerPoint и OneNote. Хотя Office 365 является платным коммерческим продуктом, работать с Web Apps можно легально и бесплатно через SkyDrive — запущенное в 2007 г. облачное хранилище, синхронизация файлов с которым происходит через одноименное десктоп-приложение. Теперь SkyDrive позиционируется в качестве хранилища файлов, созданных в онлайн-офисе.

В 2012 г. Google представил облачное хранилище Google Drive, которое стало местом для хранения файлов, созданных в Google Documents and Spreadsheets. До этого создавать онлайн-документы можно было только на сайте Google Documents, теперь же файлы можно создавать на локальном компьютере, после чего происходит синхронизация с drive.google.com.

Эти два решения в своем базовом варианте бесплатны для любого пользователя и, следовательно, могут быть широко использованы в процессе

обучения соответствующим технологиям. При этом существуют также:

• облачные хранилища, не связанные с каким-либо редактированием документов, но бесплатные в базовом варианте (например, Dropbox);

• облачные решения, связанные с каким-либо онлайн-приложением, но предполагающие платную подписку (например, Adobe Creative Cloud);

• приложения, позволяющие редактировать документы тех или иных типов и сохранять их в каком-либо из существующих облачных хранилищ (например, ABBYY FineReader Online или CodeAnyWhere в интеграции с DropBox; Splash-Up в интеграции с Flickr);

• приложения, существующие как надстройки или "плагины" (расширения) для облачных решений (например, LucidChart для Google Drive);

• облачные решения, предоставляющие доступ к виртуальным операционным системам и виртуальным серверам (например, Microsoft Windows Azure или Selectel.ru).

Для простоты мы будем считать все вышеперечисленные варианты "облачными решениями", поскольку даже простое приложение-редактор работает в связке с хранилищем, и все процессы происходят в Интернете, предположительно с участием множества различных серверов.

С точки зрения веб-разработки большая часть современных возможностей облачных решений обеспечивается со стороны клиента технологиями HTML5, CSS, Javascript+AJAX. В браузер загружается некоторая стартовая веб-страница, которая содержит весь интерфейс взаимодействия с пользователем (теперь такой подход известен под названием Single Page Application). Если в недавнем прошлом обогащенные интерфейсы и работа с мультимедиа были возможны только на основе Flash (Flex), то сейчас все эти задачи решаются с помощью HTML5 и современных мультимедийных форматов (H.264, WebM, Theora и т.п.).

Приведем теперь конкретные примеры. Целесообразно использовать метод проектов, ставя задачу, решение которой требует проведения микроисследования. Уровень сложности микроисследования можно варьировать: "продвинутым" учащимся можно предлагать выбор и обоснование выбора облачных инструментов; учащимся с более слабой входящей подготовкой можно предложить готовое решение и инструкцию по выполнению всех необходимых действий. Инструкции целесообразно подготавливать в виде **скринкастов** (а не в виде текстовых документов или распечаток), поскольку интерфейсы облачных решений меняются часто и создавать печатные инструкции просто бессмысленно:

Пример 1. "Фоновое" использование облачных технологий при изучении любого элемента содержания:

• Часть урока с объяснением нового материала удобно проводить в форме (интерактивной) онлайн-конференции. Учитель создает документ в Google Drive и предоставляет его в публичный доступ (возможно, с разрешением редактировать). Учащиеся открывают его на своих рабочих местах. Все, что учитель печатает или редактирует в документе, мгновенно отображается в браузерах учащихся, а в случае интерактивной работы то же справедливо и в обратную сторону;

• Параллельно используются специализированные онлайн-ресурсы (ЭОР или облачные решения), ссылки на которые размещаются в основном документе. Сюда относятся и упомянутые выше скринкасты (и любые другие дидактические материалы), они размещаются в облачном хранилище с публичным или поименным доступом;

• Ссылки на все материалы хранятся на сайте или блоге учителя (портфолио учителя), а результаты работы учащихся в их аккаунтах и представляются в доступ на их сайтах или блогах (портфолио учащегося). В совокупности получается ЭОИС данного предмета.

Эта организация работы позволяет сделать обучение практически полностью мобильным. Большую часть действий учащиеся могут совершать дома, что делает переход к дистанционной форме градиентным и полностью реализует любую нужную модель электронного обучения.

Скажем, при изучении темы "Системы счисления" локальная ЭОИС включает:

• ЭОР, лекционные или интерактивные, а также электронные версии учебников, например, *book.kbsu.ru/theory/chapter4/1_4_6.html*;

• Скринкасты, демонстрирующие алгоритмы перевода, например, *j.mp/2bin-1* или *j.mp/2bin-xl-1*;

• Электронные таблицы в Google Drive, содержащие образцы перевода чисел с помощью формул, например, *j.mp/2bin-xl-2*;

• Онлайн-калькулятор для проверки правильности перевода, например, *numsys.ru* или *baseconvert.com*.

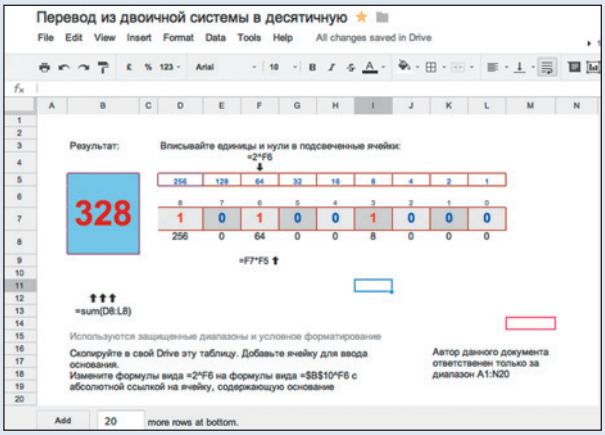


Рис. 1. Предоставленная в общий доступ интерактивная электронная таблица

Построить урок можно по-разному: например, разместившись за рабочими местами, попросить учащихся открыть скинкаст и инструкцию (по

ссылке с блога учителя), затем открыть демонстрационную электронную таблицу и скопировать ее в свой Drive, внести предложенные изменения (добавить возможность изменять основание системы счисления и создать абсолютные ссылки), затем предоставить доступ учителю к измененному документу. Затем учитель предлагает для самостоятельного выполнения учащимся индивидуальные задания (заранее подготовленные наборы чисел для перевода), результаты которого они могут представить в виде:

- электронной таблицы с общим доступом;
- скринкаста;
- флэш-ролика.

Образцы ЭОР по этой теме в виде флэш-роликов можно найти среди ресурсов к учебнику по информатике 8–9-х классов И.Г. Семакина и соавторов (§16) *school-collection.edu.ru* — эти ЭОР могут быть просмотрены непосредственно в браузере.

В каталоге ФЦИОР также доступны такие ресурсы (*fcior.edu.ru*), но, к сожалению, заявленная возможность просмотра в браузере без установки ОМС-плеера пока не реализована.

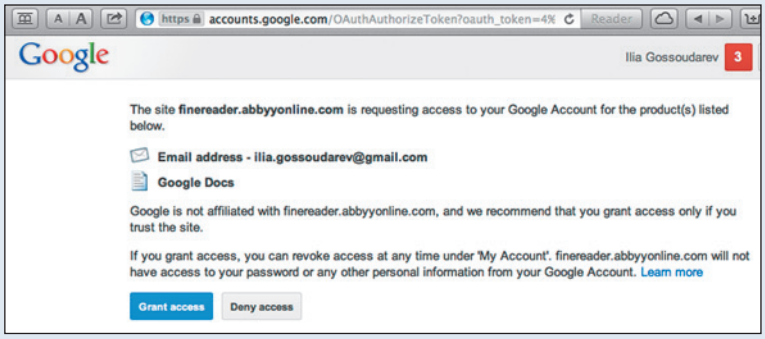


Рис. 2. Разрешение приложению вносить изменения в Drive

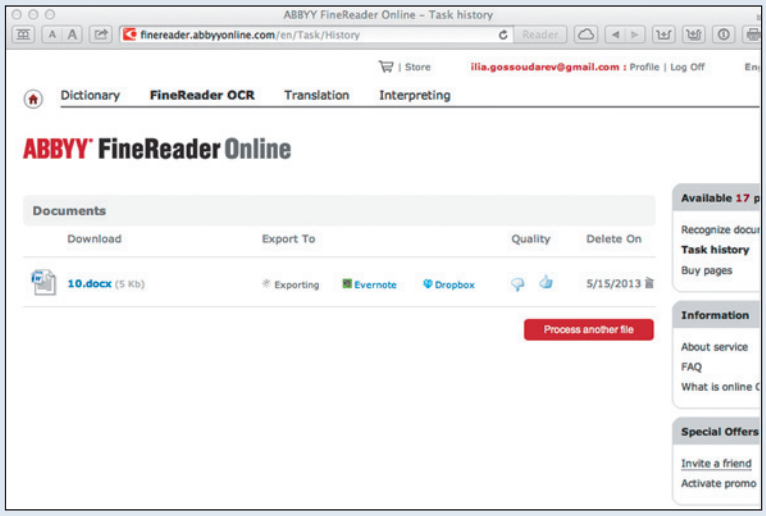


Рис. 3. Процесс экспорта из веб-приложения в облачное хранилище

Совершенно аналогично строятся уроки и по другим “вычислительным” темам, например, на решение типичных для ГИА и ЕГЭ “битовых” расчетных задач на разном содержательном материале (разрешение изображения, скорость передачи файла, глубина кодирования звука), задач на построение графов (поиск кратчайшего пути), задач на формулирование логических условий в базах данных и др.

Объединяющим свойством во всех этих случаях является высокий уровень реализации внутрипредметных связей и межпредметных связей (числа – системы счисления – электронная таблица – гиперссылка – браузер), пронизывающих изучаемую область.

Пример 2. Обучение офисным технологиям (текстовые документы, электронные таблицы, презентации) и работе с векторными изображениями.

Постановка задачи: отсканировать страницу текста со схемой (диаграммой, чертежом), распознать текст и отформатировать его по образцу, получить публичную ссылку для скачивания.

Для решения этой задачи можно использовать два инструмента:

- ABBYY Finereader Online;
- Google Drive.

Алгоритм решения:

- отсканировать страницу и сохранить в файл нужного формата;
- создать аккаунт в ABBYY Finereader Online и загрузить в него полученный файл;
- запустить распознавание, на этапе экспорта выбрать пункт Google Drive;
- разрешить приложению Finereader вносить изменения в Google Drive;
- войти в Google-аккаунт, найти в каталоге полученный документ;
- осуществить редактирование опечаток и форматирование;
- удалить из документа растровый рисунок со схемой и создать схему заново с помощью встроенного векторного редактора;
- предоставить документ в публичный доступ и отправить учителю полученную ссылку.

Даже в простом варианте решение задачи должно занять не менее трех уроков (вариант: урок с постановкой задачи + домашнее выполнение + урок с защитой полученных результатов). При решении данной типично проблемной задачи задействуется весь спектр технологий и отрабатываются метапредметные умения.

Пример 3. Обучение обработке растровых изображений.

Постановка задачи: отсканировать фотографию, изменить ее резкость, добавить декоративные элементы и подпись, получить графический файл формата PNG/JPEG, получить публичную ссылку для скачивания.

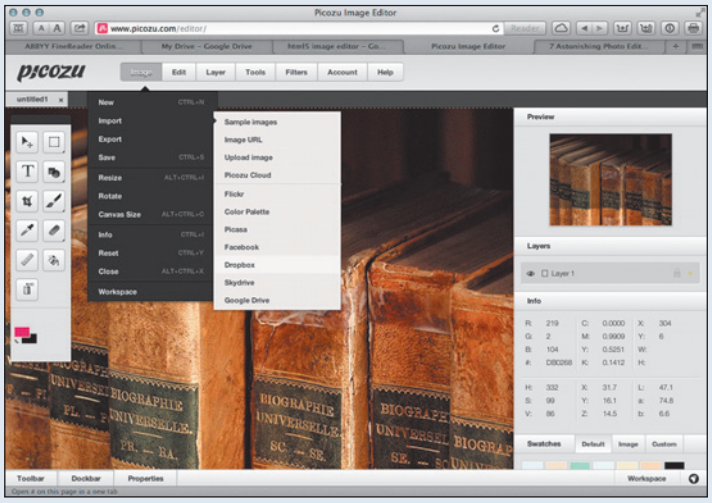


Рис. 4. Интерфейс онлайн-редактора растровых изображений

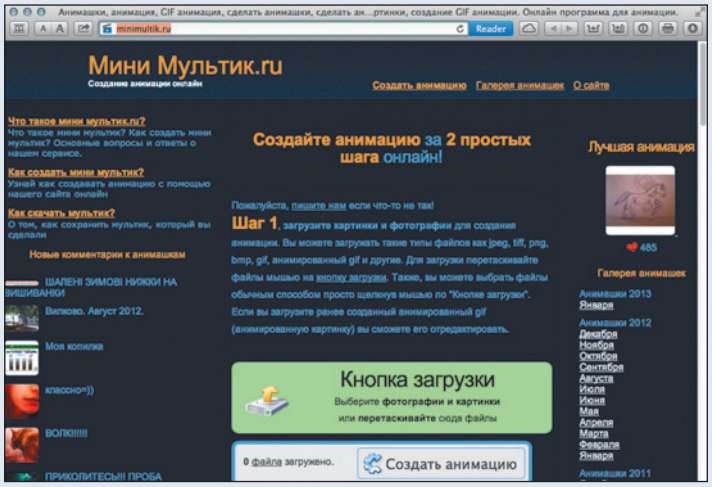


Рис. 5. Пример онлайн-редактора GIF-анимации

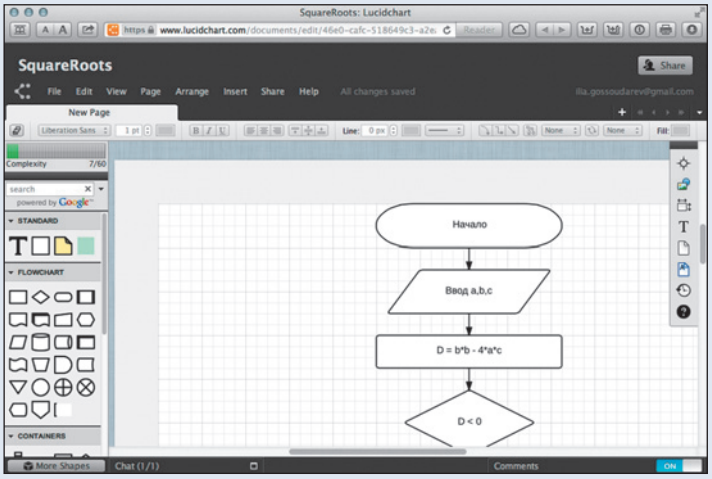


Рис. 6. Редактирование структурной схемы

Для решения этой задачи можно использовать один или два инструмента:

- Google Drive совместно с надстройкой pixlr.com;
- Splash-up (Picozu).

Алгоритм решения:

- отсканировать фотографию и сохранить в файл нужного формата;
- в Google Drive связать аккаунт с приложением pixlr.com;
- отредактировать рисунок в приложении и сохранить его снова в Google Drive;
- предоставить рисунок в публичный доступ и отправить учителю полученную ссылку.

Логичным расширением этой задачи является создание последовательности изображений, которые при быстрой смене позволяют получить иллюзию движения, т.е. создание анимации. Для этого можно использовать онлайн-редакторы GIF-анимаций, такие, как *minimultik.ru*; проект хорошо подходит для организации творческих конкурсов.

Интерфейсы графических редакторов — удобный материал для тренинга метапредметных умений, связанных с анализом и исследованием вновь изучаемых объектов. В рамках данного проекта или дополнительно можно предложить учащимся сравнить несколько редакторов по набору параметров, предоставленных учителем.

Пример 4. Обучение алгоритмизации и программированию.

Постановка задачи: по словесному описанию алгоритма создать структурную схему и написать программу (а также модифицировать ее в соответствии с индивидуальным заданием), предоставить ссылки на блок-схему и на работающую программу.

Для решения этой задачи можно использовать два инструмента:

- Google Drive совместно с надстройкой *lucidchart.com*;
- ProgrammingABC WDE.

Алгоритм решения:

- в Google Drive связать аккаунт с приложением pixlr.com;
- нарисовать структурную схему и сохранить рисунок в Google Drive;
- написать программу в ProgrammingABC или дополнить (исправить) программу, предложенную в виде ссылки учителем;
- опубликовать ссылки на схему и работающую программу.

Существует множество реализаций инструментов для программирования онлайн. Ни один из них пока не может конкурировать с мощными IDE уровня Visual Studio; варьируется набор функций и удобство работы, но базовые задачи с их помощью вполне можно решать. Веб-базированная IDE ProgrammingABC позволяет сохранять в аккаунте исходные тексты программ на языках Pascal(ABC), Python, C# (по состоянию на май 2013 г.), запускать, предо-

ставлять в общий доступ и публиковать их. Pascal реализован в этой среде с поддержкой растровой графики.

Весьма удобным онлайн-инструментом является редактор-визуализатор *pythontutor.com* — он позволяет трассировать код Python с выводом подробной информации о каждом выполняемом шаге; каждая такая визуализация может быть встроена в сайт (блог) в виде виджета на основе IFrame.

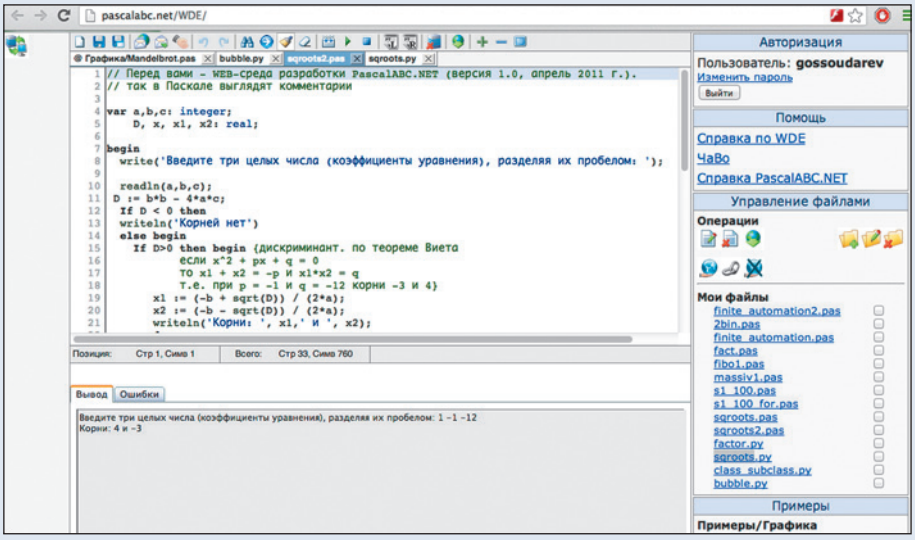


Рис. 7. Работа в онлайн-среде IDE

Для онлайн-обучения языку Ruby может быть использован интерактивный онлайн-интерпретатор *tryruby.org*, оформленный в виде книги, в страницу которой обучаемому предлагается вводить инструкции на Ruby.

Если необходимо проиллюстрировать различия в реализации каких-либо алгоритмических структур или алгоритмов на разных языках, то для этого удобно использовать параллельно два инструмента:

- мультиязыковой репозиторий реализаций алгоритмов *rosettacode.org*;
- мультиязыковой онлайн-интерпретатор/компилятор *codepad.org* (заявлена поддержка C, C++, D, Haskell, Lua, OCaml, PHP, Perl, Python, Ruby, Scheme, Tcl).

Бесплатные инструменты предоставляются “as is”, без всяких гарантий работоспособности, и это может стать препятствием при организации обучения только на их основе. Гарантированные решения всегда платны. Это касается и онлайн-систем визуального дизайна (Adobe Creative Cloud), и облачных операционных систем (Windows Azure), и рассмотренных выше онлайн-сред разработки (compilr.com), и других подобных решений. Но стоимость чаще всего достаточно ($10–20 в месяц) низка для того, чтобы имелась возможность поддерживать в школе хотя бы один платный аккаунт для демонстрации функциональности системы обучающимся.

В этой статье многие ставшие общеизвестными моменты опущены (онлайн-тренажеры слепого набора текста на клавиатуре, например), но детальная разработка вопросов обучения любой теме с помощью облачных инструментов — это, на наш взгляд, интересная задача для сообщества учителей, результатом решения которой мог бы стать каталог облачных приложений в виде wiki-ресурса.

Литературные и интернет-источники

1. Диски в облаках // Информатика, № 8, 2012. С. 10–13.

2. Amazon EC2 Beta [Электронный ресурс] URL: *http://aws.typepad.com/aws/2006/08/amazon_ec2_beta.html* (дата обращения: 01.05.2013).

3. The Grid: Blueprint for a New Computing Infrastructure [Электронный ресурс] URL: *http://www.elsevierdirect.com/v2/companion.jsp?ISBN=9781558609334* (дата обращения: 01.05.2013).

4. Федеральные государственные образовательные стандарты [Электронный ресурс] URL: *http://минобрнауки.рф/документы/543* (дата обращения: 01.05.2013).

5. Федеральный Закон об образовании [Электронный ресурс] URL: *http://минобрнауки.рф/документы/2974* (дата обращения: 01.05.2013).

6. *Лапчик М.П.* Россия на пути к Smart-образованию // Информатика и образование, 2013, № 2 (241). С. 20–26.

7. *Тихомиров В.П., Тихомирова Н.В. и др.* Россия на пути к Smart-обществу. М.: IDO Press, 2012, 280 с.

ИСТОРИЯ ИНФОРМАТИКИ

Древние… роботы

> В конечном счете рано или поздно может появиться такой робот, интеллект которого будет сравним с интеллектом среднего человека.
>
> *Виктор Михайлович Глушков, советский математик, кибернетик*

Сегодня роботами никого не удивить. Они уже используются на производстве и в быту, при чрезвычайных ситуациях и для развлечений. А экспериментальные роботы, разработанные в Японии, например, такие как Asimo, “Джеминоид Ф” и другие, просто поражают.

Рис. 1. Справа — робот-девушка

Слово “робот” придумал выдающийся чешский писатель Карел Чапек. В 1920 году он написал пьесу под заглавием “RUR” (сокращение от “***R**ossum's **U**niversal **R**obots*” — “Россумовские универсальные роботы”). Так были названы человекоподобные машины — изобретение инженера Россума, способные выполнять за человека все виды работ. Герой пьесы — главный инженер россумовских фабрик Гарри Домин — так рассказывает историю создания роботов: “Изготовление роботов основано на открытии, сделанном великим физиологом Россумом. Однажды Россум, тогда еще молодой ученый, отправился на какой-то далекий остров изучать морских животных. При этом он делал попытки воспроизвести протоплазму, пока не открыл вещество, обладающее всеми свойствами живой материи, из которого можно было образовать любое живое существо, начиная от инфузорий и кончая человеком”. Племянник изобретателя — инженер Россум-младший — решил использовать гениальное открытие, чтобы изготовлять “живые и разумные рабочие машины”. “Один робот, — сказал он себе, — с прокормом обходится в три четверти цента в год и вполне заменяет двух с половиной человек. Роботам можно читать Библию или логарифмы, кормить их ананасами или соломой — чем угодно, им это все равно, у них нет никаких потребностей и ощущений. Они не имеют воли, страстей, души. Роботы не дорожат жизнью, она им не нужна. Когда они изнашиваются, их выбрасывают”.

Далее пьеса Чапека рассказывает о том, как некая акционерная компания наладила массовый выпуск дешевых роботов, чтобы заменить ими живых людей — рабочих и служащих на фабриках и заводах, как впоследствии роботы вышли из повиновения и восстали против своих создателей.

Талантливое произведение Чапека — острая сатира на тех, кто боится техники, считая, что машина (а сегодня — и компьютер) может поработить человека. Нет нужды говорить, насколько необоснованны эти страхи. Человек всегда будет повелителем машины, он ее творец. Но слово “роботы” навсегда осталось в лексиконе человечества (сегодня человекоподобные роботы также часто называют “андроидами”).

Теперь давайте немного помечтаем: представим, что мы находимся в кабине машины времени, которая может доставить нас в далекое прошлое. Цель нашего путешествия — проследить долгий и трудный путь человека к осуществлению мечты о механическом помощнике, который мог бы облег-

чить тяжелый, изнурительный труд на производстве, помочь в домашнем хозяйстве.

При раскопках гробниц фараонов археологи обнаруживали среди них усыпальницы детей. Вместе с детьми в пирамидах были найдены куклы с подвижными руками и ногами. Это первое дошедшее до наших дней свидетельство о стремлении людей еще в давние времена сотворить искусственного человека.

Древние греки и римляне строго почитали трогательный и немного грустный обычай — прощание с детством. Вступая в пору зрелости, дети приносили своих кукол перед заходом солнца в храм богини Афродиты или Венеры. В Мексике и Перу, в пустынях Африки, джунглях Австралии археологи при раскопках также часто находили кукол.

История создания человекоподобных движущихся механизмов восходит к далекому прошлому, к тем временам, когда сказка смешивалась с былью и каждая легенда отражала мечту народа.

Фантастическая машина времени переносит нас на древний остров Крит. Ярко светит солнце. Круто поднимаются горы в заоблачную высь. Оливковые деревья раскинули свои ветви… Плодовые сады, пальмовые рощи — маленький рай посреди лазурного моря.

Много чудесных легенд можно услышать на этой земле. Одна из них рассказывает о необыкновенном доме, вызывающем удивление и восхищение всех островитян. Внешне здание напоминало поросший цветами холм, а внутри комнаты были расположены в виде лабиринта, по которому мог пройти только хозяин дома, мудрый Дедалос. Никому из посторонних не удавалось разгадать тайну великого мастера и выйти самому из необычного жилища.

Создатель этого лабиринта Дедалос прославился также изготовлением деревянных человечков, которые могли шагать и двигать руками, и особенно своим самым прекрасным творением — деревянной движущейся фигурой богини Афродиты.

Папирусы древности донесли до нас описания многих механизмов, подражающих движениям человека и животных. Легенды гласят, что тарентский философ и математик Архит в IV веке до н.э. смастерил деревянного голубя, который махал крыльями и мог даже взлетать; египетский фараон Птолемей Филадельфийский в III веке до н.э. заставил своего раба изготовить куклу, подражающую движениям человека.

Наше путешествие в глубь веков продолжается… II век до н.э. Древняя Греция. На одной из узких улочек Афин живет математик и механик Герон Александрийский[1]. Слава о нем гремит по всей стране. Его принимают жрецы храмов и императоры. Герон — непревзойденный мастер, создающий искусственных животных. Толпы народа стекаются к его дому, где каждый вечер перед заходом солнца он показывает чудесную машину. Маленькие певчие птички сидят напротив огромной совы. Как только сова отворачивается от птичек, певуньи начинают весело свистеть. Но стоит грозной сове повернуть голову к птичкам — и те сразу замолкают.

Это устройство — автомат Герона Александрийского (*рис.* 2) действовал с помощью очень остроумной конструкции. Были использованы два закрытых сосуда — большой и маленький. На крышке одного из них помещались птички, другого — сова. В большой сосуд стекала вода, постепенно вытесняя воздух, который, проходя по тонкой трубочке, издавал свистящий звук, напоминающий пение птицы.

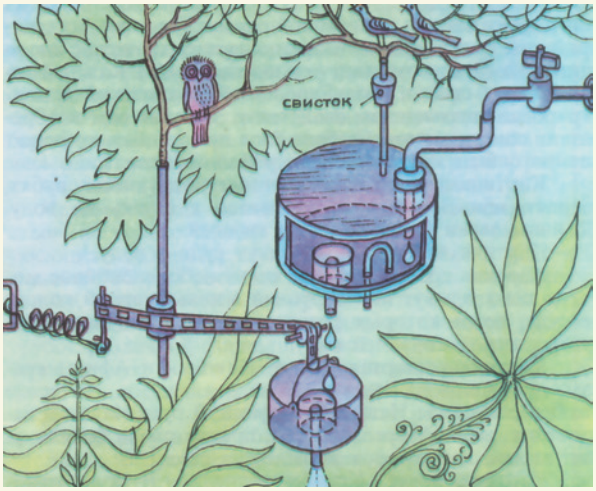


Рис. 2. Схема автомата “Сова и поющие соловьи”

Заполнив этот сосуд, вода по отсосной трубе начинала переливаться в другой, масса которого соответственно увеличивалась. С помощью веревки, перекинутой через блок, прикрепленный к валу, поворачивалась ось с сидящей на ней совой. Уровень воды в сосуде в это время был небольшим, воздух по трубке не выходил, и птички “молчали”. После того как большой сосуд освобождался, ось, а вместе с ней и сова возвращались в исходное положение. Затем процесс повторялся. Уровень воды в сосуде опять повышался, сова отворачивалась, певчие птички начинали свистеть.

Автомат Герона чрезвычайно прост. Его можно изготовить самим из совсем простых деталей (старый водопроводный кран, аквариум, латунные или алюминиевые трубки).

Машина времени не спешит покидать дом Герона Александрийского, вся жизнь которого была отдана изобретательству. Через несколько лет после создания “поющих птиц” он придумывает еще более интересный механизм — театр автоматов. О его устройстве мы знаем из книги “Автоматы”, написанной самим Героном Александрийским (так впервые в мире появилось греческое слово *автоматос*, что означает *самодвижущий*). Каждый вечер у ворот своего дома Герон Александрийский демонстрировал согражданам свой театр. Вот как сам изобретатель описывает

[1] Если вы знаете о формуле Герона для расчета площади треугольника по длинам его сторон, то именно он является ее автором.

это необычайное представление, сюжет которого был заимствован из “Малой Илиады”:

— *Картина первая*. Данайцы чинят суда, пилят, рубят, сверлят, вбивают гвозди, готовятся к спуску на воду. Слышны звуки вколачиваемых гвоздей.

— *Картина вторая*. Люди тянут суда в воду.

— *Картина третья*. Зритель видит небо, спокойное море, по нему плывут под парусами в кильватерной колонне суда, возле которых резвятся дельфины. Начинается шторм, строй кораблей нарушается.

— *Картина четвертая*. Месть Навплия и Афины грекам, побившим их сына Паломеда камнями. На сцене стоят герои пьесы Навплий и Афина. В руке Навплия зажигается факел. Мореплаватели, приняв огонь факела за свет маяка, направляют корабли на скалы.

— *Картина пятая*. Кораблекрушение. В волнах появляется плывущий Аякс, слышится удар грома. В руке Афины сверкает молния, поражающая Аякса. Аякс исчезает, исчезает также и Афина. Представление заканчивается.

Театр Герона поражал присутствующих: механические фигуры разыгрывали сложные действия, занавес открывался автоматически, так же происходила смена декораций.

На первый взгляд механические игрушки могут показаться всего лишь забавой, однако они стоят в самом начале пути, по которому шло развитие автоматов.

Рис. 3. Герон Александрийский

Механические чудеса

Тридцать долгих лет — с 1216 по 1246 год день за днем трудился талантливый немецкий философ и алхимик Альберт Великий, чтобы построить своего железного человека. Трудно описать механическое чудо того времени. Механический человек, одетый в ливрею, “служил” в доме Альберта Великого привратником. Железный слуга открывал дверь и приветствовал входящего поднятием руки. Мастер так хорошо сделал своего “привратника”, что его трудно было отличить от настоящего человека. И вот однажды любимый ученик Альберта Великого — епископ Фома Аквинский, которому до этого не приходилось видеть “привратника”, пришел в гости к своему учителю. Дверь ему открыл железный слуга. Вид “привратника”, его лицо и фигура смутили епископа. “Нечистая сила!” — воскликнул Фома Аквинский и стал громить механизм тяжелым посохом. Когда на шум прибежал Альберт Великий, все было кончено. От “привратника” остались одни шестеренки, каркас и винты. Чертежи, расчеты и описания чудной машины не сохранились.

Может быть, это всего лишь легенда, но она говорит о том, что человек всегда мечтал создать механическое “существо”, не только внешне напоминающее его самого, но умеющего говорить, слушать, думать. Уже в XIII веке епископ Роберт Гросетест пытался сделать “говорящую” голову, которая могла бы произносить отдельные слова. Предложенная им схема состояла из сложной системы трубок различной формы и длины, через которые продувался воздух, в результате чего возникали звуки, отдаленно напоминающие человеческий голос.

Эпоха Возрождения оставила миру не только шедевры искусства, но и образцы интересных машин и механизмов. Гениальный ученый и изобретатель Леонардо да Винчи сконструировал и построил сложную механическую модель льва.

Дворец короля Людовика XII в Милане. Тронный зал. На троне восседает сам король. Открывается дверь, и в зал входит огромный лев, замирают пораженные придворные. Страшный зверь подходит к трону, останавливается у ног Людовика XII, поднимает лапу, открывает отверстие в груди, и оттуда падают белые лилии — эмблема французских королей.

Продолжая наше путешествие на машине времени, перенесемся в Средневековье, когда человек создает часовой механизм — одно из самых совершенных устройств в мире. Именно часы — первый автомат, созданный человеком, первое программное устройство. Завод пружины приводит в действие сложнейшие шестеренки и зубчатые колеса, рычаги и кулачковые механизмы. Часы как бы получают задание произвести ряд определенных движений: стрелки движутся по кругу, звенит колокольчик и т.д.

С помощью часового механизма можно открывать и закрывать дверцы, заставлять двигаться фигурки животных, как это сделано на фасаде Центрального театра кукол им. С.В. Образцова в Москве.

Рис. 4

Часовой механизм стал основой для подавляющего числа автоматов, имитирующих живые существа, созданных в Средние века и чаще всего служащих движущимися деталями, приводимыми в действие механизмом часов.

Первые башенные часы были установлены в 1352 году на Страсбургском соборе. В первоначальном виде они не сохранились, до нас дошло только их описание. Механизм указателя времени в них занимал три этажа, на каждом из которых во время боя часов появлялись фигуры, изображающие людей или животных. Утро начиналось кукареканьем петуха. Статуя, возникающая на нижнем этаже, указывала дату. На втором этаже по кругу проходили фигуры, выполненные в виде знаков Зодиака, фаза их обращения точно соответствовала действительному периоду движения зодиакальных созвездий. На самом верху через каждый час появлялись три библейских царя и степенно кланялись.

Слава об этих часах гремела по всему миру, они считались главной достопримечательностью страны. Отовсюду приезжали люди полюбоваться необыкновенным чудом. Но шли годы, механизм старел и ветшал. В 1547 году часы пришлось остановить для ремонта. И лишь через тридцать лет, в 1574 году, вторично рожденные, они были пущены в ход. Венгерский путешественник Мортон Чомбор, посетив в 1620 году Германию, с восторгом отозвался об этих часах.

Через два столетия страсбургские часы вновь подверглись реконструкции. Часовой механизм стал приводить в движение ряд символических фигур. Каждый час разыгрывался маленький спектакль, изображающий течение жизни человека: в первые 15 минут появлялась фигурка младенца, через четверть часа выходил юноша, еще через 15 минут — зрелый муж, а по истечении часа проходил, ковыляя, старик в сопровождении смерти с косой за плечами. Ровно в полдень на террасе, где смонтированы часы, появлялись аллегорические фигуры и раздавался крик петуха.

Диковинные часы, прошедшие через века как памятник, обессмертивший человеческий труд, до сих пор являются одной из главных достопримечательностей города Страсбурга. Сейчас эти часы работают уже после третьей реконструкции.

Машина времени переносит нас в XIII век. Не следует забывать, что развитие науки и техники тогда еще только начиналось.

Плеяда гениальных ученых того времени занималась созданием и конструированием автоматов. Кабинет средневекового философа Альбертуса Магнуса. Сидя за большим столом, хозяин принимал гостей. Входные двери с почтительным поклоном открывал… механический человек.

Знаменитый средневековый астроном Иоганн Регионтан (1436–1476), живший в Германии, много лет трудился над созданием механического орла. Дошедшие до наших дней описания свидетельствуют, что когда император Священной Римской империи Максимилиан II Габсбургский въезжал в Нюрнберг, механический орел приветствовал его наклоном головы и хлопаньем крыльев.

И снова встреча через века. Дворец французского короля Карла V. Минуя парадные залы и пышные гостиные, спустимся по винтовой лестнице в полуподвальное помещение, где придворный часовщик Джуанелло Турриано изготовляет заводные игрушки в виде животных и шагающих человечков. Особенно удалась ему шагающая лошадь, которая брыкалась и прыгала, когда на нее садился “наездник”.

Достижения отдельных талантливых самоучек быстро распространяются по миру. Начинается повальное увлечение механикой — конструируются часы с пастухом и пастушками, танцующими дамами, львы и собаки разыгрывают сцены, и все это управляется часовыми механизмами.

Даже в наши дни, проходя по залам Эрмитажа, мы не устаем восхищаться произведениями талантливых мастеров-механиков, которые вручную, пользуясь совсем простым инструментом, изготовляли часы с движущимися фигурками, исправно работающие и поныне.

Наша машина времени направляется в XVIII век, который был назван современниками “веком часов”. Петербург. Российская Академия наук. В те годы механическими мастерскими заведовал Иван Петрович Кулибин (1735–1818) — талантливый изобретатель, создавший много любопытных механизмов, в том числе и знаменитые часы, хранящиеся в Эрмитаже. Часы эти по внешнему виду и величине напоминали гусиное яйцо. В золотом корпусе художественной работы кроме часового механизма находился миниатюрный театр автоматов, где крохотные фигурки разыгрывали сцену, сопровождаемую перезвоном. Сигналом к началу представления был поворот специальной стрелки. Ровно в полдень часы играли гимн, а в течение второй половины суток вызванивали мелодию, сочиненную самим изобретателем. Каждые час, полчаса и четверть часа отмечались особым перезвоном.

В XVII–XVIII веках в России было немало и других мастеров-умельцев, проявлявших чудеса изобретательности и таланта. Лучшие русские мастера начинают успешно конструировать всевозможные механические устройства, приводимые в движение часовыми пружинами. Газеты пестрят объявлениями о демонстрациях таких устройств. Вот, например, что писала газета “Санкт-Петербургские ведомости” в № 59 за 1777 год: “С дозволения главной полиции показываема здесь будет между Казанской церковью и Съезжей в Марковом доме прекрасная, невиданная здесь никогда механическо-музыкальная машина, представляющая изрядно одетую женщину, сидящую на возвышенном пьедестале, играющую на поставленном перед нею искусно сделанном флигеле (клавесине) десять отборнейших, по новому вкусу сочиненных пьес, то есть три

менуэта, четыре арии, два полонеза и один марш. Она с превеликой скоростью выводит наитруднейшие рулады и при начатии каждой пьесы кланяется всем гостям головою. Искусившиеся в механике и вообще любители художества немало будут иметь увеселения, смотря на непринужденные движения рук, натуральный взор ее глаз и искусные повороты ее головы, все сие зрителей по справедливости в удивление привести может".

Имена талантливых русских изобретателей Кулибина, Ползунова, братьев Черепановых, о которых много написано в литературе, пользуются мировой известностью. Однако в отечественной истории техники много незаслуженно забытых имен. К ним можно отнести имя талантливого русского изобретателя XIX века Антона Марковича Гамулецкого, который всю жизнь занимался конструированием автоматических устройств. В газете "Известия" № 130 (18585) рассказывалось, что "…в 1794 году в сорокалетнем возрасте Гамулецкий возвращается в Россию и поступает на службу. В 1808 году в чине коллежского регистратора уходит в отставку и целиком отдается любимому делу: он создает различные неведомые дотоле механические и физические приборы и автоматы".

Проходят годы неутомимого труда изобретателя. В изобретения вкладываются все небольшие сбережения. Царское правительство не помогало талантливому изобретателю. Но воля и настойчивость победили. Через двадцать лет работы Антон Маркович открыл "механический кабинет" для всеобщего обозрения. Успех был огромный. Любознательные посетители часами ждали очереди, чтобы посмотреть, как тогда называли, "чудеса механики". И чудеса действительно начинались у входа. На верхней площадке лестницы посетителей встречала парящая в воздухе фигура, выполненная в рост человека. Экскурсанты сами могли убедиться, что механический человек не был подвешен сверху и не имел подпорок снизу или с боков. Как только посетитель вступал на последнюю ступеньку, "летающий человек" поднимал руку, в которой держал валторну, и приветствовал вошедших звуками торжественной музыки.

"Десять лет, — пояснял Гамулецкий, — я трудился, чтобы найти точку опоры и вес магнита и железа, дабы удержать фигуру в воздухе…".

Далее "Известия" так сообщают о чудесах, которые творились в механическом кабинете: "Едва посетитель усаживался на диван, установленный в кабинете, как начинала звучать приятная музыка — внутри дивана автоматически включался особый музыкальный ящик. Затем гость брал в руки "волшебную палочку" и, взмахивая ею, сам совершал "чудеса"; по его желанию на полу начинала разъезжать маленькая колесница, из ваз, расставленных по углам, выскакивали называемые посетителями карты… Гамулецкий всегда сам демонстрировал свои изобретения. С особой гордостью изобретатель показывал купидона, оттачивающего стрелу, амура, играющего на арфе, железного петуха, вдохновенно кричащего "кукареку", лающую механическую собачку, черную кошку, которая мяукала, лениво выгибая спину, змею, с шипением проползающую через салон. Наибольшее удивление вызывала говорящая голова "чародея", отделанная под бронзу".

Судьба этих автоматов, к сожалению, неизвестна. До нас не дошли ни схемы, ни конструктивные решения интересных механизмов Гамулецкого.

Особую страницу в истории создания механических автоматов открыл французский механик Жак Вокансон (1709–1782). Созданная им утка и поныне слывет среди специалистов самым удивительным аппаратом такого рода. Прежде чем мы узнаем о том, как работает утка Вокансона, давайте познакомимся с самим изобретателем.

Жак де Вокансон жил в Гренобле и работал инспектором шелковой мануфактуры. Он изобрел первый в мире автоматический ткацкий станок, много сделал для родного города, соорудил в нем водопровод. Но не только этим запомнился Вокансон соотечественникам. В историю механики он вошел как отец латунной утки и железного флейтиста.

Вокансон построил механическую утку в натуральную величину. Эта утка, повторяющая до мельчайших подробностей живой образец, щелкала клювом, плавала, брызгалась в воде, двигала крыльями, чистила и расправляла перья, крякала, вытягивала голову вверх, выклевывала из протянутой ладони зерна и глотала их, повторяя глотательные движения живой утки, и даже "переваривала" корм (*рис*. 5).

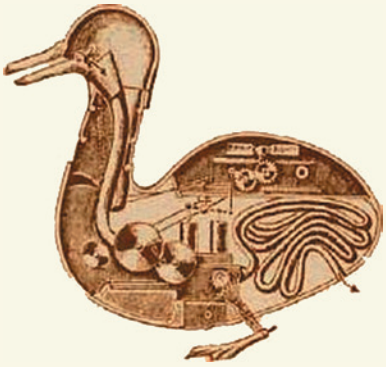


Рис. 5. Разрез механической утки Вокансона

Сложным был жизненный путь этой утки. Она украшала ряд выставок, французская Академия после смерти Вокансона оспаривала у представителей шелковой мануфактуры право на обладание чудесной птицей. Несколькими годами позже она совершила путешествие по странам Европы как главный экспонат выставок, и, наконец, ее за большие деньги приобрел для своего "кабинета искусств и чудес" Готтфрид Кристоф Бейрейс, профессор естествознания, химии, физики и медицины при университете в Хельмштадте. Все свое состояние этот человек тратил на приобретение технических диковинок.

В 1805 году, совершив продолжительное и трудное путешествие, к профессору Бейрейсу постучался иностранец, который из всех его сокровищ заинтересовался только уткой. К тому времени птица уже была почти “развалиной”, ее искусственное перьевое одеяние было порядком подпорчено молью. Но, несмотря на это, посетитель в том же году написал большую статью, посвященную чудо-птице, ради которой он предпринял путешествие из Веймара. Статья принадлежала государственному министру Иоганну Вольфгангу Гёте.

После Хельмштадта утка еще раз появляется в Париже, а затем ее следы теряются в неизвестности. Никто не знал, за какую цену она меняла хозяев, но это должны были быть колоссальные суммы.

Спору нет, утка Вокансона остается техническим чудом, но мастер создал целый ряд других, не менее интересных работ, таких, например, как механический флейтист, законченный в 1738 году. Музыкант имел рост 178 см и играл на флейте, вдувая воздух с помощью мехов и перебирая клавиши пальцами в определенной последовательности. Он исполнял одиннадцать различных мелодий. Конструктор долго изучал игру на флейте, чтобы заставить своего флейтиста безошибочно двигать пальцами с помощью обычных пружин. Везде, где демонстрировался флейтист, собирались восторженные толпы. Наконец Жак де Вокансон решил показать свое создание членам французской Академии наук. На заседании разгорелись бурные страсти. Игра флейтиста была настолько хороша, что члены Академии заподозрили автора в обмане. Чтобы реабилитировать себя, Вокансон написал брошюру, в которой подробно объяснил устройство своего изобретения.

Благодаря нашей фантастической машине времени перенесемся теперь в Австрию XVIII века.

Большое оживление царило во дворце королевы Марии-Терезии. Придворные обменивались сенсационной новостью: Фаркаш Вольфганг Кемпелен собирался демонстрировать свой шахматный автомат-машину, по словам изобретателя, оставившую позади себя все, что было создано человеком. Однако у этого события есть своя история — несколько лет назад венгерский инженер-механик Кемпелен из города Пижани заверил королеву, что сможет создать автомат, который… будет думать. В теплый майский вечер 1769 года он демонстрировал это изобретение.

Открываются большие двери, и на четырех колесиках в зал вкатывают чудо-машину размером 120 × 80 см. Снимают покрывало. Перед собравшимися предстает странная большая кукла, одетая турком, сидящая на комоде (*рис.* 6 и 7). На коленях у турка шахматная доска. Кемпелен открывает дверцы комода, и взорам присутствующих открываются валы, шестеренки, трансмиссии, пружины. Дверцы закрываются. Начинается игра. Королева делает свой первый ход. Затем угловатым, дрожащим движением делает ответный ход турок. Шахматная баталия разгорается, в конце ее происходит что-то невероятное. Губы турка издают звук, похожий на слово “шах”. Еще несколько ходов — и королева проигрывает партию. Гул изумления и восторга прокатывается по залам дворца. Королева считалась хорошим игроком… После этой победы начинается триумфальное шествие чудесного автомата по всему миру. Сенсационному турку проигрывают великий князь Павел, Фридрих II, Наполеон и Екатерина II. Постепенно следы этой машины теряются. Стало известно, что Кемпелен расстается со своим автоматом, однако время от времени турок появляется в разных городах: в 1818 году — в Милане и Париже, в 1821 году — в Лондоне.

Рис. 6. Екатерина II и “турок”.
Кадр из фильма “Шахматист” (1926 г.) [2]

Каким же образом все-таки приводилась в движение эта удивительная чудо-машина? Проникнуть в ее тайну помог сам изобретатель. Перед смертью Вольфганг фон Кемпелен делает сенсационное разоблачение. Он заявляет шокированной публике, что его машина приводилась в действие не только механизмом, но и… человеком, спрятанным внутри комода, на котором сидел турок, и управлявшим механизмом аппарата. Уже значительно позднее стало известно, что многие знаменитые шахматисты того времени за огромные вознаграждения играли с высокопоставленными особами, исполняя роль турка.

Сегодня большая часть историков считает, что эта конструкция была такова [2]. Игрок, сидевший внутри ящика с шахматной доской, не был виден даже при открытых дверцах, расположенных в передней части ящика. Игрок не имел возможности видеть положение фигур на доске, поэтому Кемпелен разработал специальную систему сигнализа-

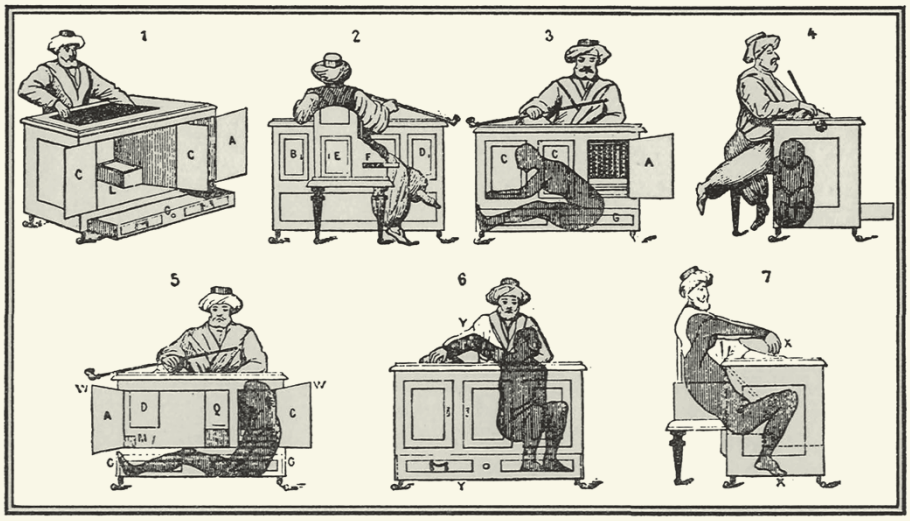


Рис. 7. “Турок” [2]

ции. В основание каждой шахматной фигуры был вмонтирован сильный магнит. Внутри ящика, под каждым полем шахматной доски, помещался надетый на тонкий стержень металлический шарик. Когда противник поднимал фигуру, шарик опускался по стержню вниз. Если на это поле ставилась другая фигура, шарик притягивался ее магнитом и поднимался вверх. Таким образом спрятанный игрок узнавал о сделанном противником ходе. Кроме того, он управлял сложнейшей системой рычагов, приводивших в движение руки и голову турка, а также захватывавших и переставлявших фигуры.

Эта система управления была тщательно замаскирована. Но, кроме нее, в ящике размещалась еще одна сложная конструкция из зубчатых колес, шестеренок и рычагов, которая предназначалась для показа публике. Перед началом партии фон Кемпелен большим ключом заводил механизм и приводил его в действие. Во время партии он повторял эту процедуру после каждых двенадцати движений аппарата: с одной стороны, это поддерживало в публике иллюзии самостоятельной работы автомата, а с другой, — давало спрятанному игроку время отдохнуть и обдумать позицию. Можно также отметить, что механизм работал довольно-таки шумно. Это также было сделано намеренно, с тем, чтобы по возможности заглушить звуки, которые мог произвести спрятанный оператор. Для пущего эффекта фон Кемпелен разработал также специальные ритуалы — например, при объявлении шаха турок трижды кивал головой, а если противник делал неправильный ход, застывал в неподвижности до тех пор, пока тот не исправлял свою ошибку.

Возникает справедливый вопрос: почему никто не мог увидеть спрятанного игрока, когда Кемпелен перед началом каждого сеанса открывал дверцы комода для обозрения? Дело в том, что никогда все дверцы не открывались одновременно, это делалось в определенной последовательности, чтобы шахматист, находящийся в комоде, успел сменить свое местоположение и при этом остаться невидимым. Иллюзию пустоты создавали зеркала, расположенные под соответствующими узлами, а также специальные перегородки. И только когда все дверцы закрывались, игрок выпрямлялся и начинал манипулировать различными механизмами и рычагами.

Вот такая была “думающая машина”, перед которой даже короли снимали шляпу.

“А шахматы?” — спросите вы. Настоящий шахматный автомат построил в 1890 году испанский инженер Торрес Кеведо. Изобретенное им механическое устройство было довольно простым, оно разыгрывало окончание партии — так называемый “ладейный эндшпиль” (король и ладья против короля). Человек играл королем, и машина всегда выигрывала партию. О других шахматных автоматах можно прочитать в интересной книге [3].

В этой же книге описаны “говорящие” и подобные устройства, издающие звуки. С такими устройствами связывают одну не совсем обычную историю.

Рассказывают случай, произошедший с механическим человеком, который “умел” играть на трубе. Его построил немецкий изобретатель Иоганн Кауфман. В 1806 году Наполеон отдыхал во дворце после битвы при Иене. Вдруг среди ночи сон его был прерван звуком трубы. Это был сигнал прусской кавалерии. Поднялась тревога. Однако вскоре выяснилось, что кто-то, проходя по темным залам дворца, нечаянно нажал пусковую кнопку искусственного трубача.

Продолжим наше путешествие в мир механических автоматов и перенесемся в XVIII век, в Швейцарию. В маленькой деревушке Шо де Фон, расположенной возле самой французской границы, живут отец и сын. Оба часовщики. Отец — Пьер-Жак Дро много сил и труда вкладывает в обучение своего сына любимой профессии Часовых Дел Мастера. Далеко за полночь сидят отец и сын, разбирая устройства сломанных часовых механизмов. Надо отдать должное сыну Анри — он чрезвычайно способный ученик, старается во всем быть похожим на отца, работой которого очень гордится. Слава

о талантливых мастерах далеко разнеслась по всей Швейцарии и Франции.

Успех, выпавший на долю отца и сына Дро, был завоеван ими упорной, кропотливейшей работой. Пьер-Жак Дро создал множество интересных механизмов. Достаточно вспомнить прекрасные маятниковые часы, сконструированные им для короля Испании Фердинанда IV. В часы был встроен механизм, который управлял фигурами пастушка и собаки. “Когда часовая стрелка подходила к какому-нибудь часу, — пишет известный советский популяризатор науки О.Дрожжин, — пастушок подносил ко рту флейту и свистел столько раз, сколько должно быть пробито часов. У ног пастушка лежала собачка, охраняя корзину с яблоками. Если кто-либо пытался взять яблоко, механическая собачка начинала лаять. Когда яблоко клали обратно, собака на часах замолкала”.

Однако самую большую известность принесли отцу и сыну Дро механические люди. Целые толпы путешественников спешили в маленькую деревушку, чтобы посмотреть сделанного Пьером-Жаком Дро механического человека, который умеет писать (*рис.* 8).

Рис. 8. Андроид — мальчик-писец отца и сына Дро

Во второй половине XVIII века этому действительно трудно было поверить. Но молва оказалась права. В 1770 году Пьер-Жак Дро заканчивает своего механического писца. За работой отца внимательно следил его шестнадцатилетний сын Анри. Через четыре года после ее окончания он создает собственную конструкцию — механического художника. Затем оба умельца начали постройку механической музыкантши.

Шел 1774 год. В Париже открылась выставка, на которую отец и сын Дро представили свои изобретения. Выставка имела огромный успех. Многочисленные зрители бурно приветствовали механических собратьев.

Что же представляли собой механические люди отца и сына Дро? Вот как их описывает О.Дрожжин в книге “Разумные машины”: “Писец был ростом с пятилетнего ребенка; он сидел на скамейке перед столиком. В правой руке маленького механического человека было гусиное перо (в то время стальных перьев еще не знали). Писец макал перо в стоящую перед ним чернильницу и писал разные слова и даже фразы без всякого участия человека. Буквы были крупные, красивые, с нажимом и располагались в ровные строчки. Между словами он оставлял промежутки. При писании механический человек двигал головой и, казалось, следил за тем, что пишет. Окончив работу, писец посыпал лист бумаги песком для высушивания чернил, а потом стряхивал его. Другой механический человек, таких же размеров, как и первый, держал в руке карандаш и рисовал разные фигурки. Рисовал не сразу, а с остановками, как бы размышляя. Иногда дул на лист, чтобы удалить соринки. Рисунки получались удачные. Музыкантша — тех же размеров, что и два ее “брата”, — играла на фисгармонии[2], ударяя пальцами по клавишам. Четко и легко удавались ей трели и быстрые пассажи. Она поворачивала также голову и глаза, как бы следя за положением рук. Ее грудь поднималась и опускалась, будто дышала. Окончив игру, механическая женщина наклоняла голову, благодаря слушателей за одобрение”.

Рис. 9. Андроид — девушка-музыкантша

Движения всех трех механических людей были так естественны, что многие зрители готовы были считать их живыми. И только когда Дро, открывая дверцы со стороны спины, показывал находящийся внутри сложный механизм, зрители начинали верить, что перед ними действительно произведения техники, а не живые существа.

“Железные люди” Пьера и Анри Дро получили название “андроиды”, что означает “человекоподобные, человекообразные”. В некоторых книгах, посвященных роботам, авторы пишут, что слово “андроид” образовалось от имен и фамилии отца и сына Дро (Ан-дро-ид), другие склонны считать, что это производное от искаженного греческого слова “антропос” — “человек”.

Успех на Парижской выставке натолкнул Пьера Дро на мысль показать своих механических людей испанскому королю Фердинанду IV. Писец, рисовальщик и музыкантша были погружены на парусник, и путешествие началось, но возле берегов Испании корабль потерпел крушение. Анри Дро и его механические люди хотя и вымокли, но были

[2] Клавишный духовой музыкальный инструмент, по форме напоминающий небольшое пианино, по звучанию — орга́н.

спасены. Морская вода — страшный враг металла, он начинает быстро ржаветь, механические люди были повреждены. Анри разобрал все механизмы, почистил их и снова собрал. Андроиды обрели вторую жизнь. В Мадриде открылась выставка, устроенная Анри Дро. Но на этом не кончились злоключения изобретателя и его творений. Святейшая инквизиция усмотрела в работах Анри колдовство и арестовала его, изъяв механических людей. В те далекие и страшные времена обвинения в колдовстве были равносильны смертному приговору. Инквизиция пытала изобретателя и на несколько долгих лет отправила его в тюрьму…

Преодолевая неимоверные трудности, Анри вырывается из заключения и бежит на родину. Здесь в 1790 году он узнает о смерти отца и через год умирает сам. Однако для андроидов продолжается "жизнь", полная приключений…

Принцип работы андроидов был основан на использовании механической силы пружин. Создание знакомых и простых для нас механизмов для механиков прошлого было нелегкой проблемой. Вспомним, сколько шестеренок было в моделях Вокансона, Дро и других мастеров Средневековья. Сейчас эти шестеренки делаются почти мгновенно с помощью всего лишь одной операции — штамповки. Мастера Средневековья вытачивали их вручную, затрачивая на это многие часы, дни, недели, пользуясь примитивными шаблонами и напильником.

Еще более трудной была задача проектирования кинематических механизмов. Сейчас они рассчитываются по формулам, известным из теории машин и механизмов, используются также типовые проектные решения. Всем этим конечно же не могли воспользоваться Вокансон и Дро, им приходилось все делать впервые, полагаясь на мастерство и интуицию.

Шли годы, и человечество овладело новыми видами энергии. В жизнь вошли паровая турбина Джеймса Уатта, лампы накаливания Эдисона и Ладыгина. Люди вписывают новую страницу в увлекательную книгу истории. К тому времени прочно входят в жизнь электричество, радио, автоматика. Конструирование человекоподобных автоматов продолжается, но уже на новой основе. В движение их приводит электричество, они снабжаются имитаторами голоса (динамические громкоговорители), органов зрения (фотоэлементы), слуха (микрофоны и усилители низкой частоты). Поэтому приглашаем читателей в новое путешествие, в страну, где живут уже не механические, а "люди с другим источником энергии".

Первым, кто встретится на нашем пути, будет паровой человек, сконструированный в 1893 году Дж. Муром. Приводимый в действие паровой машиной мощностью 0,5 лошадиных сил, он ходил по кругу со скоростью 14 км/ч (*рис.* 10). Его одежда напоминала латы рыцаря. Шлем закрывал глаза и лоб. Во рту он держал дымящуюся сигару (таким образом из механизма выходил пар). Рост парового человека достигал двух метров. Это был, пожалуй, первый и последний паровой робот, поражающий воображение современников.

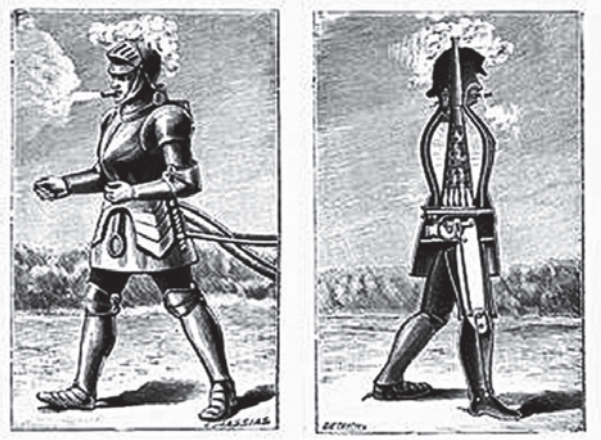


Рис. 10. "Паровой человек" Дж. Мура

А вот электрических роботов было построено великое множество, и первым из них был "Мистер Телевокс", изобретенный американским инженером Венсли. Управлялся "Телевокс" с помощью свистка. Внутри робота был установлен микрофон, который улавливал свистки-команды, а также усилитель для преобразования слабых сигналов микрофона. Сигналы поступали на вход трех избирательных фильтров, каждый из которых был настроен на определенную частоту. Затем выделенная частота включала исполнительное реле, приводящее в движение мотор.

В схему "Телевокса" был введен распределитель, который, переключаясь, коммутировал один из пяти электродвигателей.

Что мог "Телевокс?" Вот что рассказано об этом роботе в уже упоминавшейся книге О.Дрожжина "Разумные машины". "Сидя за столиком, Венсли взял в руки телефонную трубку с аппарата, стоящего перед ним. В этот самый момент что-то щелкнуло в другом телефонном аппарате перед "Телевоксом", и затем раздалось жужжание: "з-з-з-з". По-видимому, "Телевокс" отвечал, что слушает. Тогда Венсли поднес к губам свисток и издал протяжный звук: "ту-утут". "Телевокс" снова ответил, но на этот раз прерывистым жужжанием: "дз… дз… дз". Венсли снова свистит. Теперь "Телевокс" отвечает уже действием: он поднимает флаг, открывая для обозрения портрет Джорджа Вашингтона, первого американского президента.

Присутствовавшие в лаборатории Венсли журналисты и сотрудники газет разразились аплодисментами, приветствуя первое проявление способностей "Телевокса". После этого, давая различное число повторных свистков, Венсли заставил "Телевокса" пустить в ход вентилятор, зажечь лампы в комнате, открыть окно, закрыть двери, пустить в ход пылесос. Присутствующих особенно поражало то обстоятель-

ство, что “Телевокс” выполнил еще несколько приказаний, не сходя с места, не двигая ни одной частью своего деревянного нескладного тела…”.

“Смешная человекообразная внешность “Телевокса” в его действиях никакой роли не играет. Мой робот, если отбросить его оболочку, представляет собой центральную автоматическую телефонную станцию, к которой в качестве абонентов присоединено несколько электромоторов. Эти электромоторы и производят те действия, которые вы только что видели”, — говорил Венсли.

Впоследствии “Телевокс” работал дежурным при водопроводной системе одного нью-йоркского небоскреба. Он следил за уровнем воды, пускал в ход насос и т.п. “Телевокс” имел огромный успех.

Начинается изобретательская лихорадка. Создаются электрические роботы, которые “умеют делать все”. В 1928 году английским инженером Ричардсом был сконструирован робот “Эрик” (*рис*. 11). 15 сентября 1928 года этот робот выступал на ежегодной выставке общества инженеров-механиков. Удивляя всех присутствующих, он произносит длинную речь. “Эрик” обладал представительной внешностью. Он был закован в серебристые латы, внутри его узких глаз светились электрические лампочки, на груди красовалась надпись “RUR” (помните пьесу К.Чапека?). Когда “Эрика” включали, во рту его начинали светиться зеленые лампочки, сигнализирующие, что вся электрическая схема в исправности. Робот вставал, поднимал руки, разговаривал. И это особенно поражало всех присутствующих. В то время еще не было магнитофонов и звук записывался только на патефонные пластинки.

Еще один брат “Телевокса” — робот “Альфа” — детище английского профессора Гарри Мея. Масса “Альфы” — две тонны! Голова его имела вид цилиндра, вместо глаз — очки-пластины со множеством отверстий. Вместо ушей — большие микрофоны. Робот “Альфа” более совершенен, чем его “братья” “Телевокс” и “Эрик”. Он вставал, садился, поднимал и опускал руки. По просьбе зрителей двигал пальцами. Если в протянутую руку робота вставить пистолет, он очень метко стрелял вверх и вперед. На расстоянии двадцати метров все пули попадают в “яблочко” мишени. “Альфа” мог говорить, свистеть, петь, во время пения он открывал рот. Управлялся он человеческим голосом.

Успех “электрических людей”, как их называла пресса тех лет, привлек внимание электротехнических фирм, которые начинают изготовлять роботов для рекламных цепей. Так, робот “Вилли”, построенный в 1934 году компанией “Вестингауз”, мог вставать, садиться, двигаться вперед-назад и, кроме того, был заядлым курильщиком ☹.

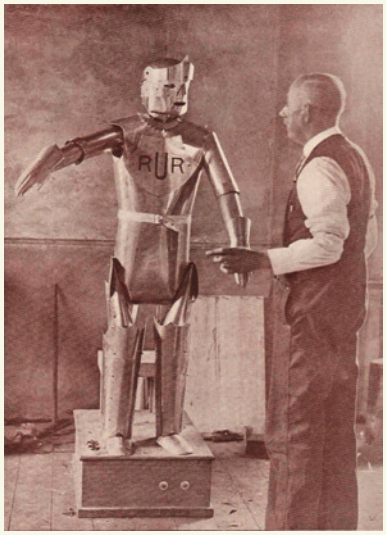


Рис. 11. Робот “Эрик”

В этом же году был построен робот, изображающий женщину, сидящую на диване. На коленях у нее лежал небольшой струнный музыкальный инструмент — цитра. Если кто-нибудь из присутствующих произносил название популярной песни, робот тотчас начинал наигрывать на цитре ее мелодию. Конечно, такая жестко запрограммированная система не могла иметь большой объем памяти, поэтому число исполняемых песен было очень ограниченным.

На Лондонской радиовыставке 1932 года тоже не обошлось без “механического” человека, который сообщал точное время и читал вслух газету (статьи из утренних газет оперативно записывались на грампластинки).

Продолжим наше путешествие и пройдемся по Всемирной выставке 1933 года, открывшейся 1 июня в американском городе Чикаго с целью показать достижения науки и техники за прошедшие сто лет. Здесь экспонировался робот, который умел продевать нитку в иголку. Он “питался” электрической энергией и сжатым воздухом и получал команды по радио.

В отделе “Медицина” демонстрировался оригинальный робот. Это был тщательно одетый четырехметровый мужчина, который читал посетителям лекцию о процессе пищеварения. Во время чтения он элегантным жестом расстегивал жилет, и зрители видели часть грудной клетки и живота, закрытые прозрачным целлулоидом. Очень хорошо просматривались пищевод, желудок, кишечник, печень. Робот водил пальцем по своим внутренностям и подробно объяснял, как работает пищеварительный тракт. Лекция продолжалась двадцать минут.

В отделе животноводства экспонировалась необычная корова. Она дышала, жевала, двигала головой вправо и влево, махала хвостом, закрывала и открывала глаза, мигала и даже… давала настоящее молоко (которое было налито в доильную установку). Этот робот был точной копией настоящей коровы. Его программный механизм был выполнен в виде вала, на котором располагались кулачки. Вал приводился в движение электромотором. Кулачки были связаны с рычагами, а те, в свою очередь, — с различными частями тела: головой, глазами, хвостом. Впоследствии механическая корова использовалась как пособие для изучения зоологии.

На Всемирной выставке 1937 года в Париже был показан сложнейший робот “Профессор Аркадиус”. Его словарный запас достигал 180 слов. За плату он составлял “психологические характеристики” и вручал их посетителям.

Описанные роботы были созданы более 70 лет назад. Читая о них, мы можем только снисходительно улыбаться. Достижения современной кибернетики и электроники позволяют строить механизмы гораздо более сложные, чем роботы двадцатых–тридцатых годов XX века. Однако нельзя забывать, что они являются прародителями современных роботов.

Литература

1. *Гордин А.Б.* Занимательная кибернетика. М.: Радио и связь, 1984.

2. *Шилов В.В.* Игры, в которые играли автоматы. Информационные технологии, 2009, № 8.

3. *Шилов В.В.* Удивительная история информатики и автоматики. М.: ЭНАС, 2011.

ЗАДАЧНИК

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

Нашего полку прибыло!

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

— учеников гимназии № 3 г. Самары, учитель **Корженко Виктория Викторовна**;

— учащихся школы № 5 станицы Платнировская Кореновского р-на Краснодарского края, учитель **Лоднова Елена Вадимовна**;

— учеников школы № 44 г. Рязани, учитель **Марцинкевич Елена Евгеньевна**.

Желаем им успехов!

15 ребусов по информатике (опубликованные в январском выпуске)

Ответы (соответствуют номерам ребусов)

1. Сервис. 2. Правка. 3. Вставка. 4. Формат. 5. Диаграмма. 6. Гистограмма. 7. Заливка. 8. Рамка. 9. Граница. 10. Формула. 11. Функция. 12. Данные. 13. Столбец. 14. Строка. 15. Лист.

Ответы прислали:

— Аджоян Кристина и Бобровская Диана, 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.**;

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

— Аношкин Иван, Евтюхин Максим, Конобеев Илья, Карташов Иван, Муравьев Иван, Симонян Роман и Чухин Павел, г. Рязань, школа № 44, учитель **Марцинкевич Е.Е.**;

— Базанов Илья, Бойцов Никита, Головченко Тихон, Тананаева Анастасия, Тананаева Ксения и Телегин Дмитрий, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

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

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

— Браславский Даниил и Николаенко Константин, Краснодарский край, Кореновский р-н, станица Платнировская, школа № 5, учитель **Лоднова Е.В.**;

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

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

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

— Вольский Михаил, г. Самара, гимназия № 3, учитель **Корженко В.В.**;

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

— Егоров Григорий, Ильина Дарья, Мышкина Мария, Никитина Мария и Скворцова Виктория, Караклинская средняя школа, Чувашская Республика, Канашский р-н, учитель **Антонова Л.В.**;

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

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

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

— Комов Александр, Нижнеломовский филиал Пензенского государственного университета, Пензенская обл., Нижнеломовский р-н, село Верхний Ломов, преподаватель **Соснина Л.В.**;

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

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

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

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

— Маршалов Владислав, гимназия г. Кобрина, Республика Беларусь, учитель **Шугай А.А.**;

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

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

Задание "Пять вопросов" (рубрика "Поиск информации")

Ответы

1. Ходжа Насреддин, чтобы заставить ишака есть щепки, надел животному очки с зелеными стеклами.

2. Из президентов США нынешний президент Барак Обама снялся в дневном телевизионном токшоу под названием "ВИД" на телеканале ABC. Кроме того, бывший президент Билл Клинтон снялся в эпизодической роли (самого себя) в фильме "Мальчишник в Вегасе-2".

В ряде ответов указывался 40-й президент США Рональд Рейган. Но он, актер по профессии, будучи президентом, в кино и шоу не снимался.

3. В карты рояль выиграла певица Клавдия Ивановна Шульженко.

4. В сериале "Игра на выбывание" главного героя едва не сожрал тигр.

5. "Красота дворцов превосходит все, что знает Париж", — так написал о Москве французский писатель Стендаль.

Ответы представили:

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

— Акиньшина Екатерина и Хомякова Анна, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Аксененко Ирина, Баков Анатолий и Чумаков Илья, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

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

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

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

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

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

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

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

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

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

— Янушкина Виктория, Москва, гимназия № 1530, учитель **Пастухов О.А.**

Отметим ответ Степана Яковлева, студента Канашского педагогического колледжа, который, кроме источников информации, привел подробные комментарии к ответам.

Головоломка "12 спичек"

Напомним, что необходимо было, используя 12 спичек, получить многоугольник площадью 4 (приняв длину спички за единицу длины).

Решение показано на фотографии, которую прислал Евгений Мороз, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1 (учитель **Васнина О.В.**).

На ней изображена фигура, основанная на так называемом "египетском треугольнике" — прямоугольном треугольнике, катеты которого равны 3 и 4, гипотенуза — 5, а площадь — 6. В этом треугольнике три спички переставлены, в результате чего площадь фигуры стала равна 4.

Кроме Евгения, ответы представили:

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

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

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

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

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

— Круглова Светлана, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

— Левченко Станислав, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

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

— Смоляков Никита, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

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

В ряде ответов получен не один, а два многоугольника.

Кроссворд (опубликованный в январском выпуске)

Ответы

По горизонтали: 7. Информатика. 8. Секунда. 9. Позиция. 14. Три. 15. Диаграмма. 16. Дек. 17. Курсив. 18. График. 19. Ять.

По вертикали: 1. Цифра. 2. Шпион. 3. Линейка. 4. Правка. 5. Заступ. 6. Таблица. 8. Строка. 9. "Наири". 11. Замер. 12. Ячейка. 13. Шрифт.

Ответы представили:

— Акиньшина Екатерина и Хомякова Анна, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

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

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

— Григоренко Василий, Григоренко Дмитрий, Круглякова Мария и Тихоненко Дарья, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

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

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

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

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

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

— Тарасов Игорь, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

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

— Устюгов Николай, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Шейкин Александр, средняя школа села Ириновка, Новобурасский р-н Саратовской обл., учитель **Брунов А.С.**

Головоломки сан-го-ку

Ответы

№ 1

5	6	9	20
7	8	4	19
1	3	2	6
13	17	15	

№ 2

1	3	5	9
2	7	8	17
4	6	9	19
7	16	22	

№ 3

7	4	5	16
1	3	6	10
2	8	9	19
10	15	20	

Правильные ответы прислали:

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

— Ахматгалиева Диана, Баширова Кристина, Ботова Дарья, Димакова Арина, Евдокимова Мария, Заляева Карина, Костылева Юлия, Мацишина Валерия, Нестерова Анастасия, Одинцова Екатерина, Танасюк Артем, Терёшкина Даша, Чаусов Евгений и Шелуханова Татьяна, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

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

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

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

— Иванова Полина, Коршаков Вадим, Олешова Дарья, Сафронов Максим и Станкевич Александра, Смоленская обл., г. Демидов, школа № 1, учитель **Кордина Н.Е.**;

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

— Кабанов Виталий, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Комов Александр, Нижнеломовский филиал Пензенского государственного университета, Пензенская обл., Нижнеломовский р-н, село Верхний Ломов, преподаватель **Соснина Л.В.**;

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

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

— Кубко Андрей, 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.**;

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

— Смирнов Александр и Чаицкий Алексей, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

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

Отметим ответ учеников из школы № 1 г. Демидова, приведших обоснование своих решений.

Головоломки "судоку" (опубликованные в январском выпуске)

Ответы представили:

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

— Ахматгалиева Диана, Задорина Наталья, Клокова Анастасия, Костылева Юлия, Нестерова Анастасия, Одинцова Екатерина, Танасюк Артем, Терёшкина Дарья и Шелуханова Татьяна, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

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

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

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

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

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

— Марков Алексей и Яснов Федор, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Новикова Анна и Потапова Алевтина, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

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

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

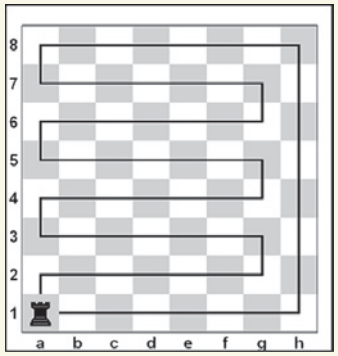
— Шаманов Петр, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**

Ответы на задание "Прилипшая монета" (начало его решения было приведено в рубрике "Крепкий орешек") представили:

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

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

Артем Танасюк решил также головоломку "Ладья обходит шахматную доску" (в ней требовалось провести ладью, перемещая ее по правилам шахмат, из некоторого поля доски так, чтобы на каждом поле фигура побывала один только раз, после чего оказалась на исходном поле). Решение показано на рисунке:



Возможны также аналогичные решения, начинающиеся с других "угловых" полей доски.

Редакция решила наградить Артема дипломом. Поздравляем!

Разбор решений числового ребуса «"СТАЙКА" из шести "ПТИЧЕК"» и задачи "Кучки монет" будет проведен в августовском выпуске "В мир информатики".

ВНИМАНИЕ! КОНКУРС

Итоги конкурса № 99

Напомним, что в качестве задания этого конкурса предлагалось выполнить ряд заданий, связанных с так называемым "полимино" — односвязными фигурами, состоящими из квадратов (односвязность фигуры означает, что каждый входящий в нее квадрат имеет по крайней мере одну сторону, общую с другим входящим в нее квадратом):

Участниками конкурса являлись:

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

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

— Гильмутдинов Ильгизар, Республика Татарстан, Актанышский р-н, село Актаныш, школа № 1, учитель **Галиева Р.Ф.**;

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

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

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

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

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

Ответы

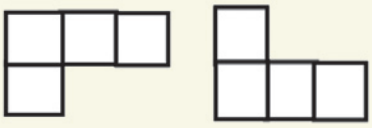
1. Количество разновидностей полимино, состоящих из:

1) четырех квадратов (тетрамино), равно 5;

2) пяти квадратов (пентамино) — 12;

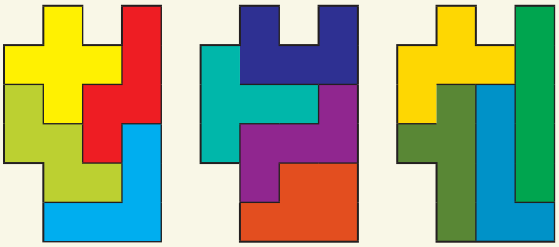
3) шести квадратов (гексамино) — 35.

При этом учитывалось, что, например, тетрамино вида:



— считаются как одна разновидность.

2. Из всех разновидностей пентамино, разделив их на три группы, можно составить три такие одинаковые фигуры:



Абсолютным победителем конкурса признан Ильгизар Гильмутдинов из Актанышской школы № 1. Дипломами будут награждены также читатели из школы № 6 станицы Барсуковская Ставропольского края, представившие ответ на второе задание (в нем каждая из трех одинаковых фигур получена из двух пентамино), Сергей Новиков из средней школы села Сердар и Артем Танасюк из школы № 124 г. Челябинска.

ЭТО ПОЛЕЗНО ЗНАТЬ

О чем рассказали старые кассеты

Ульяна Кузнецова,
ученица гимназии № 1530
г. Москвы

В нашей семье сохранился старый пленочный фотоаппарат. Остались также кассеты от пленок для него. На кассетах я увидела черные и белые квадраты (см. *рис*. 1) и заинтересовалась их назначением.

Рис. 1

Оказывается, с помощью этих квадратов закодирована информация о типе фотопленки в кассете. Причем принцип кодирования является двоичным!

Всего квадратов на кассете — 12. Их можно рассматривать как 12 битов, так как серебристый квадрат можно считать единицей, а черный — нулем.

Что означают эти биты? Дело в том, что фотопленки различаются по светочувствительности. Светочувствительность измеряется в специальных единицах, установленных Международной организацией по стандартизации (ISO), и может принимать 24 стандартных значения:

25	32	40
50	64	80
100	125	160
200	250	320
400	500	640
800	1000	1250
1600	2000	2500
3200	4000	5000

Рис. 2

Так вот, чувствительность в единицах ISO не только напечатана на упаковке, но и закодирована на кассете с пленкой.

Сколько битов требуется, чтобы закодировать все возможные значения светочувствительности пленки? Ответ прост — 5. Так как $2^4 = 16$, то четыре бита для 24 значений мало, а $2^5 = 32$ — это более чем достаточно.

Этим пяти битам соответствуют 5 квадратов в верхнем ряду, на *рис*. 3 обозначенные буквой “Ч”:

З	Ч1	Ч2	Ч3	Ч4	Ч5
З	Д1	Д2	Д3	К1	К2

Рис. 3

Таблица 1

Чувствительность	Квадрат Ч1	Квадрат Ч2	Квадрат Ч3	Квадрат Ч4	Квадрат Ч5
25	0	0	0	1	0
32	0	0	0	0	1
40	0	0	0	1	1
50	1	0	0	1	0
64	1	0	0	0	1
80	1	0	0	1	1
100	0	1	0	1	0
125	0	1	0	0	1
160	0	1	0	1	1
200	1	1	0	1	0
250	1	1	0	0	1
320	1	1	0	1	1
400	0	0	1	1	0
…	…	…	…	…	…
5000	1	1	1	1	1

Значения в квадратиках для той или иной чувствительности приведены в табл. 1.

Кроме того, в наборе квадратов закодированы количество кадров в пленке и информация об экспозиционном допуске (допустимой ошибке экспонирования).

О числе кадров “говорят” три квадратика в нижнем ряду (см. *рис*. 3), обозначенные буквой “Д”. Соответствие между длиной пленки и значениями битов показано в табл. 2.

Таблица 2

Число кадров	Квадрат Д1	Квадрат Д2	Квадрат Д3
12	0	1	1
20	1	0	1
24	0	0	1
36	1	1	0
48	0	1	0
60	1	0	0
72	0	0	0
Нестандартная длина	1	1	1

Величину экспозиционного допуска определяют два квадратика в нижнем ряду на *рис*. 3, обозначенные буквой “К”. Значения приведены в табл. 3.

Таблица 3

Допуск	Квадрат К1	Квадрат К2
±½	1	1
±1	0	1
+2 .. –1	1	0
+3 .. –1	0	0

Описанная система кодирования называется “DX-кодировкой”.

Возникает вопрос — зачем нужна эта система? Дело в том, что с ее помощью можно автоматически определять закодированные характеристики пленки.

Белые квадратики (они окрашены серебристой краской) представляют собой открытый металл и проводят электрический ток, а черные — ток не проводят.

Электрическая схема фотоаппарата построена так, что ток подводится к первому и седьмому квадратам на кассете (зеленые квадраты на *рис*. 3; они всегда серебристые). Этот ток будет (или не будет) проведен пятью контактами на квадратах со 2-го по 6-й, в зависимости от того, окрашены они серебристой краской или нет. Так, если ток присутствует на контактах 4 и 5, но отсутствует на контактах 2, 3 и 6, в фотоаппарат вставлена пленка 400 ISO (см. табл. 1). При съемке выдержка будет установлена автоматически. Аналогично автоматически определяются и другие характеристики фотопленки.

В заключение хочу предложить читателям самостоятельно определить характеристики пленок, обозначение которых приведено на *рис*. 1, а также следующих:

а)

б)

в)

От редакции. Ответы, пожалуйста, присылайте в редакцию. Фамилии всех приславших правильные ответы будут опубликованы. Срок представления ответов — 20 сентября.

Общероссийский проект

Школа цифрового века

Интернет-поддержка проекта – Издательский дом «ПЕРВОЕ СЕНТЯБРЯ»

2013/14
учебный год

Дорогие участники проекта!

Учебный год завершается.
Спасибо за плодотворное сотрудничество!

Не забудьте напомнить администратору проекта продлить участие на 2013/14 учебный год!

Технология продления участия в проекте упрощена:
коды доступа, активированные педагогическими работниками в 2012/13 учебном году,
на следующий учебный год продлеваются автоматически
(при условии, что образовательное учреждение продолжает участие в проекте)

- Срок действия проекта в следующем учебном году:
  с 1 августа 2013 г. по 30 июня 2014 г.
- Величина оргвзноса для образовательных учреждений остается прежней –
  4 тысячи рублей за весь 2013/14 учебный год
  независимо от количества педагогических работников.

НОВОЕ В СЛЕДУЮЩЕМ УЧЕБНОМ ГОДУ!

С 1 августа 2013 года:

- **расширяется линейка предметно-методических изданий –**
  новый ежемесячный журнал «Основы безопасности жизнедеятельности»
- **спецпроект: журнал «Школа для родителей»** поможет учителю сделать родителей союзниками в обучении и воспитании детей
- **каждый месяц – новый модуль** в цикле дистанционных модульных курсов «Навыки профессиональной и личной эффективности», а также вебинары по всем курсам цикла
- **бесплатный доступ к методическим брошюрам и книгам**
  «Библиотечки "Первое сентября"»

Подробности на сайте

digital.1september.ru