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

# ИНФОРМАТИКА

## 140 лет телеграфу, при помощи которого можно передавать человеческую речь

электронная ВЕРСИЯ ЖУРНАЛА в Личном кабинете на сайте www.1september.ru

издательский дом 1september.ru
Первое сентября | февраль 2016
ИНФОРМАТИКА Подписка на сайте www.1september.ru или по каталогу «Почта России»: 79066 — бумажная версия, 12684 — CD-версия

## НА ОБЛОЖКЕ

**14 февраля 1876 года Александер Грейам Белл (шотландец по происхождению, живший с 1871 г. в США) подал заявку на свое изобретение — “Телеграф, при помощи которого можно передавать человеческую речь” (телефон).**

(Всего двумя часами позже заявку на “Устройство для передачи и приема вокальных звуков телеграфным способом” подал Э.Грей из Чикаго.)

Белл хорошо понимал значение и перспективы своего изобретения и уже в 1878 году так предсказывал его будущее: “Можно представить себе, что кабели телефонных линий будут проложены под землей или подвешены в воздухе, их ответвления соединят жилые помещения, магазины, фабрики и т.д. с главным кабелем, идущим на центральную станцию, где все эти провода будут соединяться в нужных комбинациях, и, таким образом, установится непосредственная связь между двумя любыми точками в городе”. И далее: “...в будущем телефонные линии соединят центральные станции разных городов, и человек, находящийся в одной части страны, сможет словесно общаться с человеком, находящимся совсем в другой ее части”.

## В НОМЕРЕ



## В ЛИЧНОМ КАБИНЕТЕ

### Облачные технологии от Издательского дома “Первое сентября”

Все подписчики журнала имеют возможность получать электронную версию, которая является полной копией бумажной. Для получения электронной версии:

1) Откройте Личный кабинет на портале “Первое сентября” (www.1september.ru).

2) В разделе “Газеты и журналы / Получение” выберите свой журнал и кликните на кнопку “Я — подписчик бумажной версии”.

3) Появится форма, посредством которой вы сможете отправить нам копию подписной квитанции.

После этого в течение одного рабочего дня будет активирована электронная подписка на весь период действия бумажной. Справки: podpiska@1september.ru или через службу поддержки на портале “Первого сентября”.


ИНФОРМАТИКА

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

http://inf.1september.ru

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

РЕДАКЦИЯ:
гл. редактор С.Л. Островский
редакторы
Е.В. Андреева,
Д.М. Златопольский (редактор вкладки “В мир информатики”)
Дизайн макета И.Е. Лукьянов
верстка Н.И. Пронская
корректор Е.Л. Володина
секретарь Н.П. Медведева
Фото: фотобанк Shutterstock
Журнал распространяется по подписке
Цена свободная
Тираж 18 000 экз.
Тел. редакции: (499) 249-48-96
E-mail: inf@1september.ru
http://inf.1september.ru

ИЗДАТЕЛЬСКИЙ ДОМ “ПЕРВОЕ СЕНТЯБРЯ”
**Генеральный директор:** Наум Соловейчик
**Главный редактор:** Артем Соловейчик
**Коммерческая деятельность:** Константин Шмарковский (финансовый директор)
**Реклама, конференции и техническое обеспечение Издательского дома:** Павел Кузнецов
**Производство:** Станислав Савельев
**Административно-хозяйственное обеспечение:** Андрей Ушков
**Педагогический университет:** Валерия Арсланьян (ректор)

**Школа цифрового века:**
facebook.com/School.of.Digital.Age

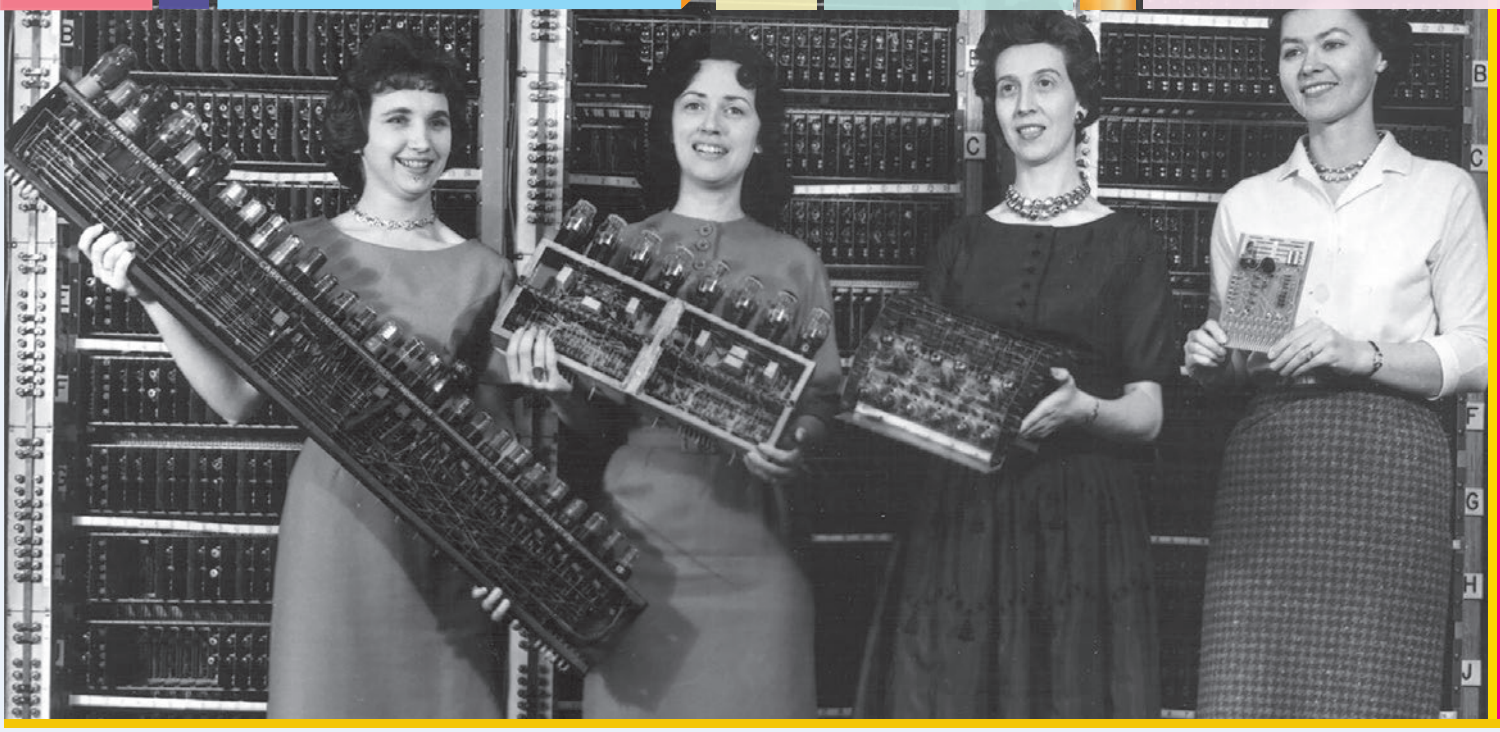
ЖУРНАЛЫ ИЗДАТЕЛЬСКОГО ДОМА “ПЕРВОЕ СЕНТЯБРЯ”
**Английский язык** – Е.Богданова
**Библиотека в школе** – О.Громова
**Биология** – Н.Иванова
**География** – и.о. А.Митрофанов
**Дошкольное образование** – Д.Тюттерин
**Здоровье детей** – Н.Сёмина
**Информатика** – С.Островский
**Искусство** – О.Волкова
**История**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

УЧРЕДИТЕЛЬ:
ООО “ИЗДАТЕЛЬСКИЙ ДОМ «ПЕРВОЕ СЕНТЯБРЯ»”
**Зарегистрировано ПИ № ФС77-58447 от 25.06.2014 в Роскомнадзоре**
Подписано в печать:
по графику 18.11.2015,
фактически 18.11.2015
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**



# Учебные задачки: 70 лет спустя

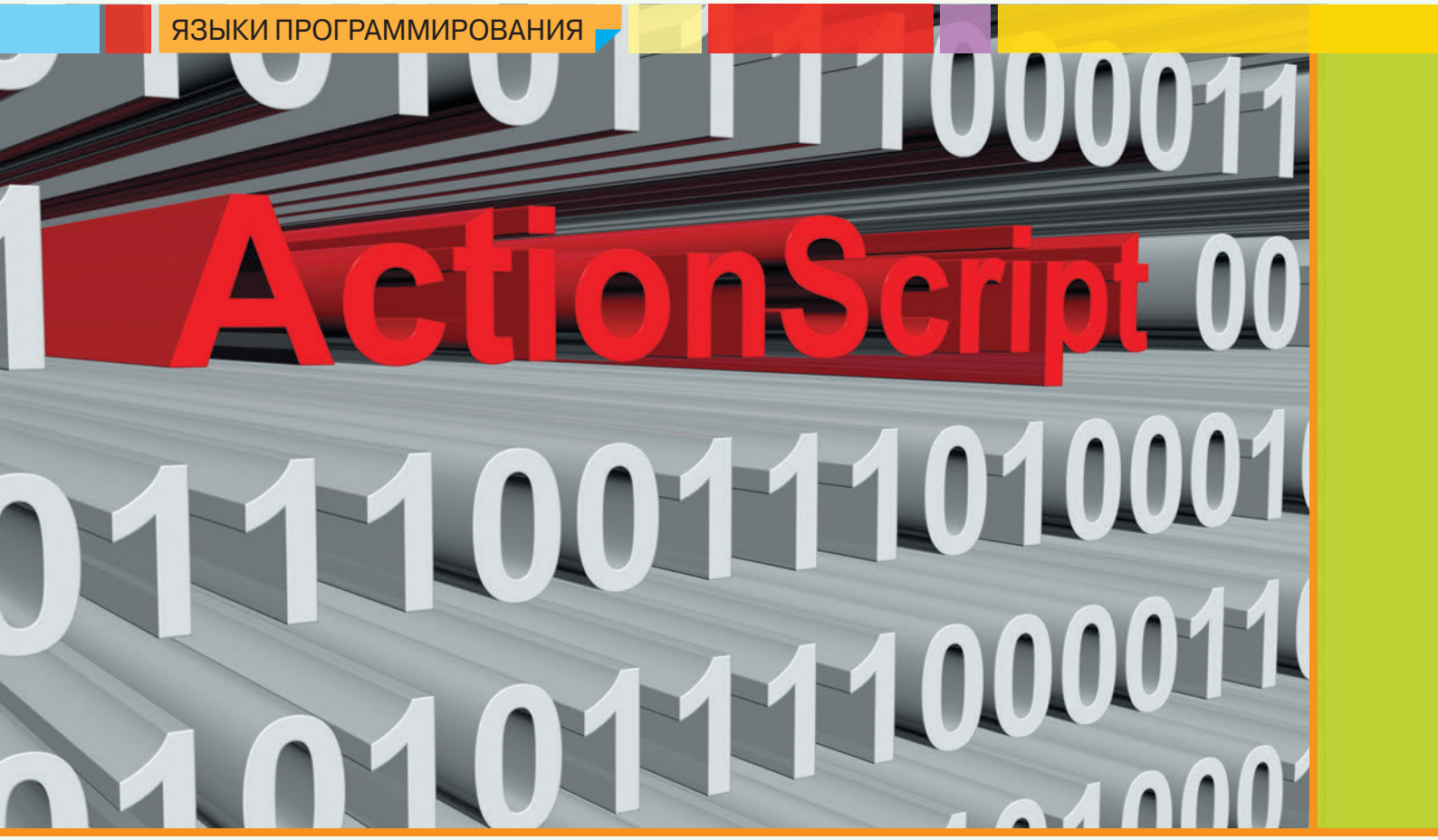
В феврале 1946 г., несколько месяцев спустя после окончания войны, широкой публике показали сверхсекретное американское оружие — компьютер ENIAC. Повоевать он, впрочем, не успел — “генералы всегда готовятся к прошлым войнам”, и разработка машины была начата лишь в 1943 г. А в июле 1944 г. прототип ENIAC получил путевку в жизнь, успешно перемножив 5 на 1000. После этого выдающегося результата (вовсе не шутка!) военные заказчики поверили, что “эта штука может быть создана”. ENIAC не был серийной машиной и, сделанный в единственном экземпляре, проработал около 10 лет, за которые сжег огромное количество энергии (хотя потреблял всего-то как 100 современных электрических чайников), переработал, отправив в утиль, бесчисленное количество вакуумных ламп, конденсаторов и резисторов.

Сейчас даже современные школьники (ну, не все и не в каждой школе) решают на уроках такие задачи, которые тогда были по силам лишь машине ENIAC. Например, в 1949 году фон Нейман использовал ЭНИАК для расчета числа $\pi$ и $e$ с точностью до 2000 знаков после запятой. Фон Неймана интересовало статистическое распределение цифр в этих числах. Предполагалось, что цифры в этих числах появляются с равной вероятностью, а значит — компьютеры могут генерировать действительно случайные числа, которые можно использовать как вводные параметры для вычислений методом Монте-Карло. Вычисления для числа $e$ были выполнены в июле 1949 года, а для числа $\pi$ — за один день в начале сентября. Результаты показали, что “цифры в числе $\pi$ идут в случайном порядке, а вот с числом $e$ все обстояло значительно хуже”. (Когда я писал эту заметку, то не удержался и написал программу для вычисления числа $\pi$ до любого знака после запятой. Саму-то программу я писал около часа — из них 40 минут искал глупую ошибку, но 2000 знаков потом получил секунд за 10…)

Упомянутый метод Монте-Карло, сегодня тоже знакомый многим школьникам, использовался на ENIAC для решения множества задач, включая моделирование погоды и течения термоядерных реакций. В силу малой мощности машины, малого объема памяти и малого времени непрерывной работы (лампы и другие устройства нещадно горели), математические модели приходилось сильно упрощать, но и при этом были получены результаты, которые в то время не могли быть получены никаким другим способом.

Славная история машины была закончена одним поворотом рубильника 2 октября 1955 г. Но в истории computer science ENIAC навсегда останется как одна из первых и выдающихся ЭВМ.

**С.Л. Островский**,
гл. редактор,
*so@1september.ru*




# Практикум программирования на ActionScript

## УРОК 12. ПОЛЬЗОВАТЕЛЬСКИЕ ФУНКЦИИ. РИСОВАНИЕ ГРАФИКА

**О.Б. Богомолова,**
**Д.Ю. Усенков,**
Москва

### Специальные имена и элементы пути

**this** — указатель на сам текущий объект, используется при вызове свойства или метода текущего объекта;

**_root** (“корень”) — начало всей иерархии объектов, используется при записи абсолютного пути.

### Глобальные переменные

Чтобы создать глобальную переменную, достаточно присвоить первому кадру анимации скрипт, содержащий команду присваивания переменной некоторого исходного значения. Далее для обращения к этой переменной достаточно в любом кадре и в любом обработчике для любого объекта использовать путь **_root.<имя_глобальной_переменной>**.

### События

**onClipEvent(load) { *программный код* }** — обработчик событий, назначаемый символу-клипу или кнопке (символу либо диалоговому элементу Button). Здесь используется событие **load** — “загрузка анимации”. Оно выполняется однократно, поэтому его программный код может быть использован для задания исходных значений переменных.

*Окончание.*
*См. № 12/2015, № 1/2016.*

## Оператор цикла с параметром

**`for (count = [исходное значение]; count > [конечное значение]; [изменение переменной count]) { программный код }`**

В первой части оператора цикла определяется переменная, которая будет меняться при каждом выполнении цикла (*переменная цикла*), и ей присваивается некоторое начальное значение. Условие во второй части оператора цикла определяет, до каких пор выполняется цикл (цикл выполняется, пока это условие истинно). Третья часть оператора цикла определяет, как переменная цикла меняется при каждом его выполнении ("проходе").

Пример:

**`for (count = 100; count > 10; count--) { программный код }`** — определяется переменная цикла с именем **count**, значение которой изначально (при первом проходе цикла) равно 100; цикл выполняется, пока значение **count** больше 10 (цикл закончится, когда значение **count** станет равно 10 или меньше); при каждом проходе цикла значение **count** уменьшается на 1 (операция **count--**).

## Условный оператор

**`if ( условие ) { программный код }`** — программный код выполняется, если условие в круглых скобках истинно (неполная форма условного оператора);

**`if ( условие ) { программный код 1 }else{ программный код 2 }`** — в зависимости от истинности условия выполняется или программный код 1 (если условие истинно), или программный код 2 (если условие ложно) (полная форма условного оператора).

Операции сравнения:

| Знак операции | Описание |
|---|---|
| < | меньше |
| > | больше |
| <= | меньше или равно |
| >= | больше или равно |
| == | равно |
| != | не равно |

**Математические методы ("функции") и математические константы**

| | |
|---|---|
| `Math.E` | Число $e$ (основание натуральных логарифмов, ≈2,718) |
| `Math.PI` | Число π (≈3,14159) |
| `Math.pow(a,b)` | Возведение в степень ($a$ — основание, $b$ — показатель, $a^b$). $a^{-m} = \frac{1}{a^m}$; $a^{\frac{m}{n}} = \sqrt[n]{a^m}$.<br>Вычисление корней из отрицательного числа не производится (возвращает сообщение об ошибке — значение **NaN** — даже для нечетной степени корня) |
| `Math.sqrt(x)` | Корень квадратный из $x$ |
| `Math.exp(x)` | Экспонента ($e^x$) |
| `Math.log(x)` | Натуральный логарифм. $\log_a N = \frac{\log_e N}{\log_e a}$ — логарифм по произвольному основанию |
| `Math.max(a,b)` | Максимальное значение из двух чисел $a$ и $b$ |
| `Math.min(a,b)` | Минимальное значение из двух чисел $a$ и $b$ |
| `Math.abs(a)` | Абсолютное значение (модуль) числа $a$ |
| `Math.round(a)` | Округление до целого по правилам математики |
| `Math.cos(α)` | Косинус угла α (в радианах) |
| `Math.sin(α)` | Синус угла α (в радианах) |
| `Math.tan(α)` | Тангенс угла α (в радианах) |
| `Math.acos(a)` | Арккосинус от $a$ |
| `Math.asin(a)` | Арксинус от $a$ |
| `Math.atan(a)` | Арктангенс от $a$ |
| Пересчет угла из градусов в радианы | $\alpha_{рад.} = \alpha_{град.} \cdot \pi/180$ |
| Пересчет угла из радиан в градусы | $\alpha_{град.} = \alpha_{рад.} \cdot 180/\pi$ |

## Динамическое рисование

Рассматриваются возможности динамического (программного) рисования векторных фигур (примитивов) в версии ActionScript 2.0 (Flash версии 8 и ниже).

> В версии ActionScript 3.0 (Flash версии 9 и выше) динамическое рисование реализовано при помощи класса **graphics**.

Рисовать можно или непосредственно в кадре (в графических командах указывается имя "**_root**"), или в отдельно размещенном на кадре объекте-клипе MovieClip (он может быть создан как вручную — нарисован и преобразован в символ MovieClip, так и программно), тогда в графических командах указывается имя "**_root.<*имя клипа*>**" либо "**this**", если команды привязаны к самому этому клипу (к событию **load**).

> Чтобы создаваемое изображение было видно на клипе, в нем обязательно должна отсутствовать фоновая закраска!

Рисовать на клипе (выполняющем функции "холста") удобнее:

— появляются возможности управления его положением в слоях и видимостью в целом;

— координаты в графических командах задаются относительно точки привязки конкретного клипа (при создании клипа точку привязки рекомендуется установить в верхнем левом углу, чтобы получить привычную "экранную" систему координат).

## Команды рисования

**.moveTo(x,y)** — поместить текущую точку ("невидимое рисующее перо") в точку с координатами *x*, *y*;

**.lineStyle(<*толщина*>, <*цвет*>, <*прозрачность*>)** — задает свойства линии рисования:

— **<*толщина*>** — толщина линии в пикселях, от 0 (невидимая линия) до 255;

— **<*цвет*>** — цвет, определяется в стандарте RGB аналогично определению цвета на web-странице: восьмеричное число, начинающееся с префикса "0x" (ноль и латинская строчная буква "икс"), в котором записаны три пары шестнадцатеричных цифр: первая пара — составляющая R (красный), вторая пара — составляющая G (зеленый), третья пара — составляющая B (синий). Каждая пара может иметь значение от 00 до FF (полная яркость). Например:

| Код цвета | Цвет | Код цвета | Цвет |
|---|---|---|---|
| 0x000000 | Черный | 0x888888 | Серый |
| 0xFF0000 | Красный | 0x880000 | Темно-красный |
| 0x00FF00 | Зеленый | 0x008800 | Темно-зеленый |
| 0x0000FF | Синий | 0x000088 | Темно-синий |
| 0xFFFF00 | Желтый | 0xFF00FF | Фиолетовый |
| 0x00FFFF | Голубой | 0xFFFFFF | Белый |

— **<*прозрачность*>** — коэффициент прозрачности: число от 0 (полная прозрачность) до 100 (полная непрозрачность);

**.lineTo(x,y)** — рисование линии от текущей точки до точки с указанными координатами (и эти координаты становятся новыми координатами текущей точки). Линия рисуется с последними установленными значениями параметров (**.lineStyle**);

**.clear()** — очистка "холста" (клипа или кадра) от всей графики, нарисованной программно (графическими командами).

Возможности рисования в версии Active Script 2.0 (Flash версии 8 и ниже) на этом исчерпываются (градиентная заливка здесь не рассматривается).

> В версии Active Script 3.0 (Flash версии 9 и выше) существует отдельный класс объектов динамического рисования: **graphics**. В нем реализовано существенно больше возможностей рисования, в том числе:
> **.graphics.moveTo(x,y)** — установка текущей точки (см. выше);
> **.graphics.lineTo(x,y)** — рисование линии (см. выше);
> **.graphics.lineStyle**(...) — задание толщины, цвета и прозрачности линий (см. выше).

## Функции пользователя

Функция (подпрограмма) — фрагмент программного кода, который можно многократно вызывать из основной программы для реализации однотипных действий с разными исходными данными.

Определение пользовательской функции в ActionScript:

```
function <имя_функции> (<переменные-параметры>) {
   <вычисления с использованием переменных-параметров>
   return <возвращаемое значение>
}
```

Переменные-параметры записывают в круглых скобках через запятую. Значение, возвращаемое через оператор **return**, может быть оформлено как переменная или выражение.

Вызов функции из основной программы (если функция объявлена как глобальная и привязана к кадру анимации):

```
_root.<имя_функции> (<фактические параметры>)
```

Фактические параметры (фактические значения переменных, которые используются при данном выполнении функции) записываются в круглых скобках через запятую в том же порядке, что и переменные-параметры в определении функции.

Вызов функции может осуществляться:

— в составе выражения, тогда возвращаемое функцией значение сразу используется в вычислении выражения;

— в правой части оператора присваивания — возвращаемое функцией значение присваивается переменной, записанной слева от оператора присваивания;

— простой вызов функции — возможен, если функция не предполагает возврат каких-либо значений (например, выводит результаты сразу в кадр анимации).

## Практические задания

### *Задание I. Рисование системы координат (график без масштабирования)*

Изображение декартовой системы координат включает в себя изображение осей *X* и *Y* со стрелками на концах, а также координатной сетки с заданным шагом.

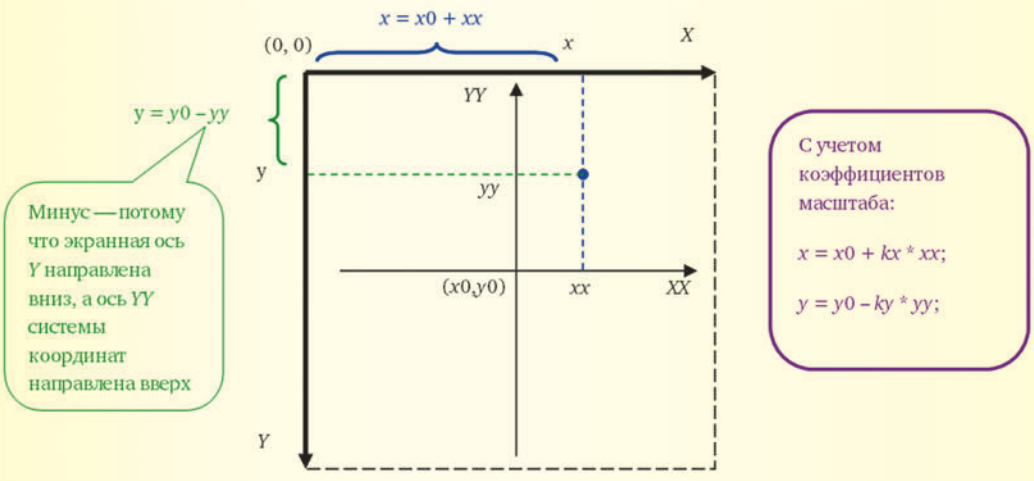
Координаты точек графика (*X* — изменение аргумента функции на некотором интервале с заданным шагом, *Y* — соответствующие значения функции) вычисляются относительно построенной системы координат. Затем полученные "относительные" координаты пересчитываются в абсолютные (относительно "экранной" системы координат, привязанной к кадру анимации либо к объекту Movie Clip, играющему роль "холста"). Удобнее всего выполнять рисование системы координат и координатной сетки не вручную, а графическими операторами, используя цикл.

Для рисования графика его начальная точка "ставится" при помощи команды **moveTo(x,y)**, а далее выполняется прорисовка отрезка от текущей точки до следующей командой **lineTo(x,y)**. Шаг изменения значения аргумента выбирается достаточно малым, чтобы ломаная линия графика выглядела более или менее гладкой.

В более сложном случае вычисляются коэффициенты масштабирования по оси *X* (зависит от ширины "холста" и от заданного диапазона изменения аргумента) и по оси *Y* (зависит от высоты "холста" и от минимального и максимального значений функции на указанном диапазоне значений аргумента). Далее получаемые "относительные" координаты точек графика пересчитываются с учетом полученных масштабных коэффициентов.

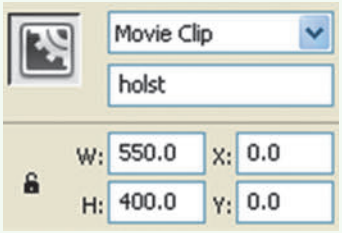
В еще более сложном варианте можно также вычислять положение нулевой точки системы координат так, чтобы обеспечить наиболее оптимальное построение графика.

Принцип пересчета исходных координат *xx* и *yy* в экранные координаты точки (*x*, *y*):



1. Создадим новую анимацию. Будем рисовать в объекте MovieClip с выводом графика сразу после запуска анимации.

2. Нарисуем в первом кадре прямоугольник с выключенным фоном. Размеры прямоугольника зададим так, чтобы он занимал максимум площади кадра анимации, но значения ширины и высоты были кратны 10. В данном случае установим размер равным 550 × 400 (размеры кадра анимации), а координаты *X*, *Y* этого объекта приравняем нулю.




Преобразуем прямоугольник в символ MovieClip (“холст”). Точку привязки MovieClip разместим в верхнем левом углу, что соответствует общепринятому положению экранной системы координат.

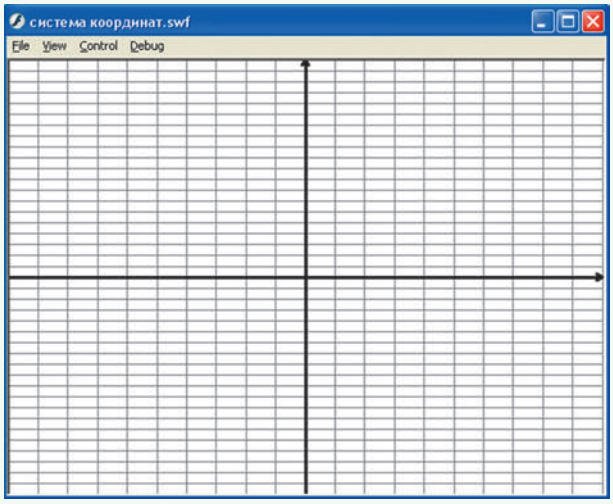
3. Присвоим первому кадру следующий программный код:

```
                    Глобальные переменные:
h0 = 400;           — высота "холста";
w0 = 550;           — ширина "холста";
x0 = w0/2;          — положение системы координат
y0 = h0/2;            на "холсте" — в его середине;
x1 = -10;           — диапазон изменения аргумента
x2 = 10;              функции — [x1,x2];
dx = 0.1;           — шаг изменения аргумента;
```

4. Присвоим объекту MovieClip (“холсту”) следующий программный код для события `load`:

```
onClipEvent(load) {                          При запуске анимации:

  this.lineStyle(1,0x888888,100);            — координатная сетка серая

  kx = (_root.w0/2) / (Math.max(Math.abs(_root.  — коэффициент масштабирования по
x1), Math.abs(_root.x2)));                   оси X

  for (xx = _root.x1; xx <= _root.x2; xx++) {  — цикл рисования вертикальных
    this.moveTo(_root.x0+xx*kx,0);             линий координатной сетки
    this.lineTo(_root.x0+xx*kx,_root.h0);
  }

  ky = 10;                                   — коэффициент масштабирования
                                               по оси Y
  for (yy = -20; yy <= 20; yy++) {           — цикл рисования
    this.moveTo(0,_root.y0+yy*ky);             горизонтальных линий
    this.lineTo(_root.w0,_root.y0+yy*ky);      координатной сетки
  }

  this.lineStyle(3,0x000000,100);            — оси черные толщиной 3 пикс.

  this.moveTo(0,_root.h0/2);                 — ось X
  this.lineTo(_root.w0,_root.h0/2);
  this.lineTo(_root.w0-5,_root.h0/2-3);      — стрелка оси X
  this.moveTo(_root.w0,_root.h0/2);
  this.lineTo(_root.w0-5,_root.h0/2+3);

  this.moveTo(_root.w0/2,_root.h0);          — ось Y
  this.lineTo(_root.w0/2,0);
  this.lineTo(_root.w0/2-3,5);               — стрелка оси Y
  this.moveTo(_root.w0/2,0);
  this.lineTo(_root.w0/2+3,5);

}
```

5. Запустим анимацию. Результат должен выглядеть так:




### *Задание II. Построение графика функции*

Построим график функции $y = f(x) = \frac{1}{1+x^2}$ .

6. Откроем ранее созданную анимацию “построение системы координат” и сохраним под новым именем (“построение графика”).

7. Программный код, привязанный к первому кадру анимации, дополним записью пользовательской функции, соответствующей математической функции, для которой требуется построить график:

```
h0 = 400;
w0 = 550;
x0 = w0/2;
y0 = h0/2;
x1 = -10;
x2 = 10;
dx = 0.1;

function f(x) {
  y = 1/(1+x*x);
  return y;
}
```

```
Глобальные переменные:
– высота "холста";
– ширина "холста";
– положение системы координат
  на "холсте" – в его середине;
– диапазон изменения аргумента
  функции – [x1,x2];
– шаг изменения аргумента;

Функция для построения графика:
- вычисление значения функции
- возврат результата
```

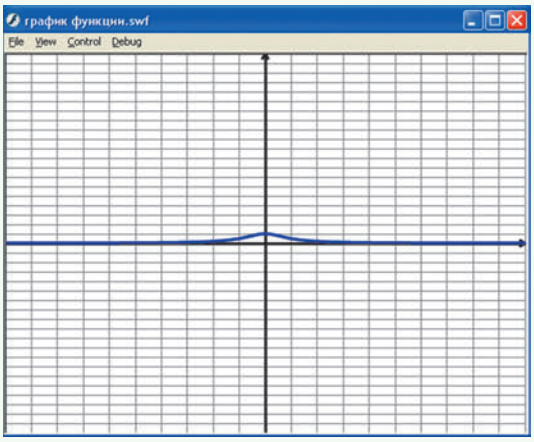
8. Программный код, привязанный к объекту MovieClip (“холсту”), дополним построением графика функции (с обращением к созданной функции пользователя):

```
onClipEvent(load) {

  this.lineStyle(1,0x888888,100);

  kx = (_root.w0/2) / (Math.max(Math.
abs(_root.x1), Math.abs(_root.x2)));

  for (xx = _root.x1; xx <= _root.x2;
xx++) {
    this.moveTo(_root.x0+xx*kx,0);
    this.lineTo(_root.x0+xx*kx,_root.h0);
  }

  ky = 10;

  for (yy = -20; yy <= 20; yy++) {
    this.moveTo(0,_root.y0+yy*ky);
    this.lineTo(_root.w0,_root.y0+yy*ky);
  }
```

```
При запуске анимации:

– координатная сетка серая

– коэффициент масштабирования по оси X

– цикл рисования вертикальных линий
координатной сетки

– коэффициент масштабирования
  по оси Y
– цикл рисования
  горизонтальных линий
  координатной сетки
```

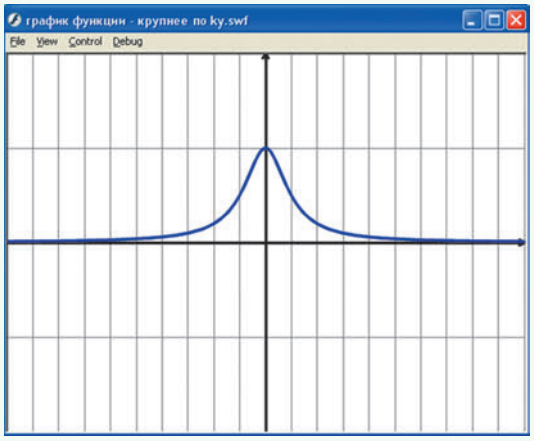
```
  this.lineStyle(3,0x000000,100);                  — оси черные толщиной 3 пикс.

  this.moveTo(0,_root.h0/2);                       — ось X
  this.lineTo(_root.w0,_root.h0/2);
  this.lineTo(_root.w0-5,_root.h0/2-3);            — стрелка оси X
  this.moveTo(_root.w0,_root.h0/2);
  this.lineTo(_root.w0-5,_root.h0/2+3);

  this.moveTo(_root.w0/2,_root.h0);                — ось Y
  this.lineTo(_root.w0/2,0);
  this.lineTo(_root.w0/2-3,5);                     — стрелка оси Y
  this.moveTo(_root.w0/2,0);
  this.lineTo(_root.w0/2+3,5);

  this.lineStyle(3,0x0000FF,100);                  — график синий толщиной 3 пикс.

  for (xx  = _root.x1; xx <= _root.x2;             — цикл перебора значений
       xx += _root.dx) {                             аргумента в диапазоне [x1,x2]
                                                     с шагом dx
    yy = _root.f(xx);                              — вычисление значения функции
    x = _root.x0 + kx*xx;                          — пересчет xx и yy в экранные
    y = _root.y0 - ky*yy;                            координаты
    if (xx == _root.x1) { this.moveTo(x,y); }      — первую точку — ставим (moveTo),
    else { this.lineTo(x,y); }                       до остальных рисуем линию
  }
}
```

9. Запустим анимацию. Проверим результат:




Для функций с разрывом (например, гиперболы или тангенса) нужно при построении графика исключать точку разрыва, программируя не один цикл изменения аргумента функции, а два или более, реализующих изменение аргумента на каждом участке непрерывности функции. Например, для гиперболы нужно построить два цикла: от **x1** до (**0 – dx**) и от (**0 + dx**) до **x2**, т.е. исключить ноль.

10. Изменим значение коэффициента **ky** с 10 на 100. Запустим анимацию, получим следующий результат:

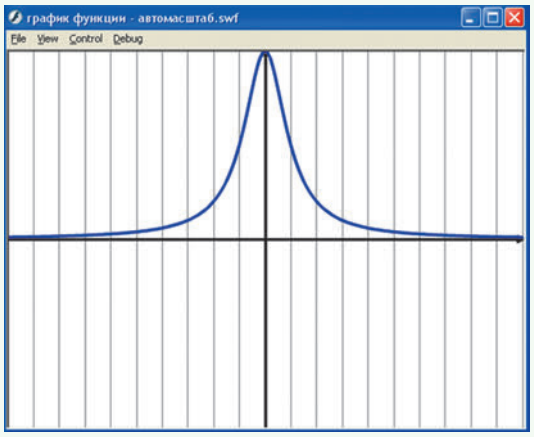


### *Задание III. Построение графика функции с автоматическим масштабированием*

11. Изменим программный код, привязанный к объекту MovieClip (“холсту”):

```
onClipEvent(load) {                                  | При запуске анимации:

  this.lineStyle(1,0x888888,100);                    | — координатная сетка серая

  kx = (_root.w0/2) / (Math.max(Math.abs(_root.x1),  | — коэффициент масштабирования
        Math.abs(_root.x2)));                        |   по оси X

  for (xx = _root.x1; xx <= _root.x2; xx++) {        | — цикл рисования вертикальных
    this.moveTo(_root.x0+xx*kx,0);                   | линий координатной сетки
    this.lineTo(_root.x0+xx*kx,_root.h0);
  }

  yymax = _root.f(_root.x1);                         | — максимум значений функции
  yymin = _root.f(_root.x1);                         | — минимум значений функции
  for (xx = _root.x1; xx <= _root.x2; xx += _root.dx) { | — перебор значений аргумента
    if (yymax < _root.f(xx))                         |   функции
    { yymax = _root.f(xx); }                         | — определение максимума
    if (yymin > _root.f(xx))
    { yymin = _root.f(xx); }                         | — определение минимума
  }
  ky = (_root.h0/2) / (Math.max(Math.abs(yymin),     | — коэффициент масштабирования
        Math.abs(yymax)));                           |   по оси Y

  for (yy = -20; yy <= 20; yy++) {
    this.moveTo(0,_root.y0+yy*ky);                   | — цикл рисования
    this.lineTo(_root.w0,_root.y0+yy*ky);            |   горизонтальных линий
  }                                                  |   координатной сетки

  this.lineStyle(3,0x000000,100);                    | — оси черные толщиной 3 пикс.

  this.moveTo(0,_root.h0/2);                         | — ось X
  this.lineTo(_root.w0,_root.h0/2);
  this.lineTo(_root.w0-5,_root.h0/2-3);              | — стрелка оси X
  this.moveTo(_root.w0,_root.h0/2);
  this.lineTo(_root.w0-5,_root.h0/2+3);

  this.moveTo(_root.w0/2,_root.h0);                  | — ось Y
  this.lineTo(_root.w0/2,0);
  this.lineTo(_root.w0/2-3,5);                       | — стрелка оси Y
  this.moveTo(_root.w0/2,0);
  this.lineTo(_root.w0/2+3,5);

  this.lineStyle(3,0x0000FF,100);                    | — график синий толщиной 3 пикс.

  for (xx = _root.x1;                                | — цикл перебора значений
       xx <= _root.x2; xx += _root.dx) {             |   аргумента в диапазоне [x1,x2]
                                                     |   с шагом dx
    yy = _root.f(xx);                                | — вычисление значения функции
    x = _root.x0 + kx*xx;                            | — пересчет xx и yy в экранные
    y = _root.y0 – ky*yy;                            |   координаты
    if (xx == _root.x1) { this.moveTo(x,y); }        | — первую точку – ставим
      else { this.lineTo(x,y); }                     |   (moveTo), до остальных рисуем
  }                                                  |   линию
}
```

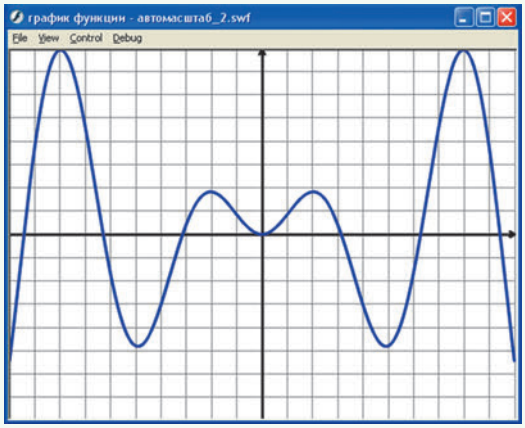
12. Запустим анимацию. Получим результат:




13. Поменяем математическую функцию на $y = f(x) = x * \sin x$. Для этого изменим запись пользовательской функции:

```
function f(x) {
  y = x*Math.sin(x);
  return y;
}
```

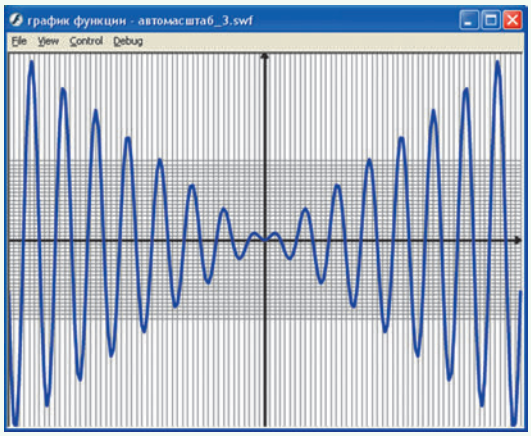
14. Запустим анимацию. Получим результат:




15. Изменим диапазон и шаг изменения аргумента функции в записях глобальных переменных:

```
h0 = 400;
w0 = 550;
x0 = w0/2;
y0 = h0/2;
x1 = -50;
x2 = 50;
dx = 0.5;
```

16. Запустим анимацию. Получим результат:



При слишком мелком масштабе координатная сетка может выводиться с дефектами, например, может быть не выведена часть линий. В этом случае можно отключить рисование сетки, вписав в программном коде в начале соответствующих графических операторов символы "//" (т.е. преобразовав эти операторы в комментарии).

Самостоятельно реализуйте возможность включения/выключения рисования сетки, например, нажатиями выбранных для этого клавиш клавиатуры.

***Подсказка:*** нужно реализовать обработку нажатия клавиш, отслеживать нажатие выбранных клавиш и менять значение глобальной переменной – флага. В основной части программного кода нужно поместить фрагмент рисования сетки в ветвь **`then`** оператора **`if`**, условие которого — это истинность переменной-флага.

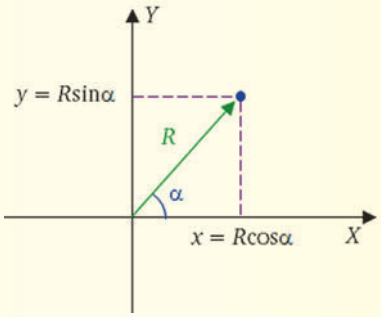
# УРОК 13. ПОСТРОЕНИЕ ГРАФИКА В ПОЛЯРНОЙ СИСТЕМЕ КООРДИНАТ

## Практические задания

### *Задание I. Рисование системы координат*

График в полярной системе координат может быть построен для функций, заданных в виде $R = f(\alpha)$, где значение радиуса (расстояния от центра системы координат) зависит от угла (ноль аргумента соответствует положительному лучу оси $X$). Далее полученные полярные координаты $(\alpha, R)$ пересчитываются в декартовы координаты $(x, y)$ по формулам:

$$\begin{cases} x = R\cos\alpha \\ y = R\sin\alpha \end{cases}$$




Изображение полярной системы координат включает в себя изображение осей $X$ и $Y$ со стрелками на концах, а также координатной сетки в виде окружностей и радиусов, выведенных с определенным шагом. При этом диапазон изменения аргумента (угла) рассматривается равным полной окружности (от 0 до $2\pi$), а диапазон изменения радиуса может быть задан изначально "с запасом" либо вычисляется исходя из максимального значения функции.

Построение графика по пересчитанным декартовым координатам $(x, y)$ точек выполняется аналогично построению графика в декартовой системе координат (см. предыдущее занятие); цикл вычисления значений функции строится для изменения аргумента (угла) на интервале $[0,2\pi]$.

1. Создадим новую анимацию. Будем рисовать в объекте MovieClip с выводом графика сразу после запуска анимации.

2. Нарисуем в первом кадре квадрат с выключенным фоном. Размеры MovieClip зададим так, чтобы он занимал максимум площади кадра анимации. В данном случае установим размер квадрата равным 400 × 400. Координату $X$ выберем так, чтобы разместить "холст" посередине кадра:


Преобразуем квадрат в символ MovieClip ("холст"). Точку привязки MovieClip разместим в середине, что будет более удобным для пересчета из полярной системы координат.

3. Присвоим первому кадру следующий программный код:

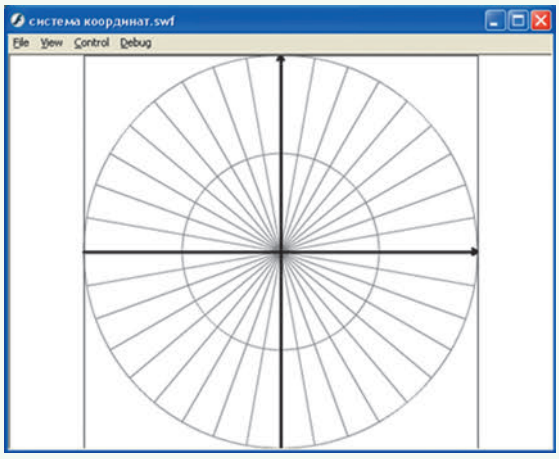
```
hw = 400;                  Глобальные переменные:
d = 0.01;                  – высота/ширина "холста";
                           – шаг изменения аргумента (угла)
```

4. Присвоим объекту MovieClip (“холсту”) следующий программный код для события **load**:

```
onClipEvent(load) {

  this.lineStyle(1,0x888888,100);

  maxR = 2;
  kR = (_root.hw/2)/maxR;

  for (rr=1; rr <= maxR; rr++) {
    for (aa=0; aa <= 2*Math.PI; aa+=0.01)
      x = rr*Math.cos(aa)*kR;
      y = -rr*Math.sin(aa)*kR;
      if (aa == 0) { this.moveTo(x,y); }
        else { this.lineTo(x,y); }
    }
  }

  for (aa=0; aa <= 2*Math.PI; aa+=10*Math.PI/180){
    x = maxR*Math.cos(aa)*kR;
    y = -maxR*Math.sin(aa)*kR;
    this.moveTo(0,0);
    this.lineTo(x,y);
  }


  this.lineStyle(3,0x000000,100);

  this.moveTo(-_root.hw/2,0);
  this.lineTo(_root.hw/2,0);
  this.lineTo(_root.hw/2-5,-3);
  this.moveTo(_root.hw/2,0);
  this.lineTo(_root.hw/2-5,3);

  this.moveTo(0,_root.hw/2);
  this.lineTo(0,-_root.hw/2);
  this.lineTo(-3,-_root.hw/2+5);
  this.moveTo(0,-_root.hw/2);
  this.lineTo(3,-_root.hw/2+5);

}
```

```
При запуске анимации:

— координатная сетка серая

— максимальный радиус
— коэффициент масштабирования

— цикл рисования окружностей
— вложенный цикл по углам
— координата x
— координата y
— первая точка ставится,
  далее рисуются отрезки

— цикл рисования
  радиусов с шагом 10 градусов
— координата x
— координата y
— рисовать от нулевой точки
  до вычисленной на max радиусе

— оси черные толщиной 3 пикс.

— ось X

— стрелка оси X

— ось Y

— стрелка оси Y
```

5. Запустим анимацию. Результат должен выглядеть так:




### *Задание II. Построение графика функции*

Построим график функции $R = f(\alpha) = \sin(4*\alpha)$ (“полярная роза”).

6. Откроем ранее созданную анимацию “построение системы координат” и сохраним под новым именем (“построение графика”).

7. Программный код, привязанный к первому кадру анимации, дополним записью пользовательской функции, соответствующей математической функции, для которой требуется построить график:

```
hw = 400;
d = 0.01;

function f(a) {
  r = Math.sin(4*a);
  return r;
}
```

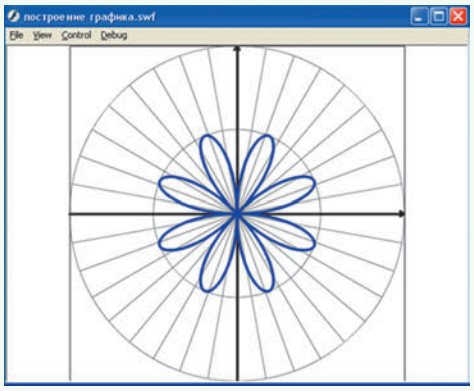
```
Глобальные переменные:
– высота/ширина "холста";
– шаг изменения аргумента (угла)

Функция для построения графика:
– вычисление значения функции
– возврат результата
```

8. Программный код, привязанный к объекту MovieClip (“холсту”), дополним построением графика функции (с обращением к созданной функции пользователя):

```
onClipEvent(load) {
  this.lineStyle(1,0x888888,100);
  maxR = 2;
  kR = (_root.hw/2)/maxR;

  for (rr=1; rr <= maxR; rr++) {
    for (aa=0; aa <= 2*Math.PI; aa+=0.01) {
      x = rr*Math.cos(aa)*kR;
      y = -rr*Math.sin(aa)*kR;
      if (aa == 0) { this.moveTo(x,y); }
        else { this.lineTo(x,y); }
    }
  }
  for (aa = 0; aa <= 2*Math.PI;
       aa += 10*Math.PI/180) {
    x = maxR*Math.cos(aa)*kR;
    y = -maxR*Math.sin(aa)*kR;
    this.moveTo(0,0);
    this.lineTo(x,y);
  }

  this.lineStyle(3,0x000000,100);

  this.moveTo(-_root.hw/2,0);
  this.lineTo(_root.hw/2,0);
  this.lineTo(_root.hw/2-5,-3);
  this.moveTo(_root.hw/2,0);
  this.lineTo(_root.hw/2-5,3);

  this.moveTo(0,_root.hw/2);
  this.lineTo(0,-_root.hw/2);
  this.lineTo(-3,-_root.hw/2+5);
  this.moveTo(0,-_root.hw/2);
  this.lineTo(3,-_root.hw/2+5);

  this.lineStyle(3,0x0000FF,100);

  for (aa=0; aa <= 2*Math.PI; aa+=_root.d) {
    rr = _root.f(aa);
    x = rr*Math.cos(aa)*kR;
    y = rr*Math.sin(aa)*kR;
    if (aa == 0) { this.moveTo(x,y); }
      else { this.lineTo(x,y); }
  }
}
```

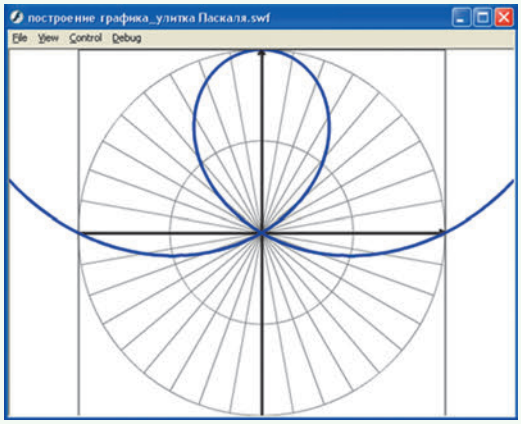
```
При запуске анимации:
– координатная сетка серая
– максимальный радиус
– коэффициент масштабирования

– цикл рисования окружностей
  – вложенный цикл по углам
– координата x
– координата y
– первая точка ставится,
  остальные – рисуются отрезки


– цикл рисования
  радиусов с шагом 10 градусов
– координата x
– координата y
- рисовать от нулевой точки
  до вычисленной на max радиусе


– оси черные толщиной 3 пикс.

– ось X

– стрелка оси X



– ось Y

– стрелка оси Y



– график синий толщиной 3 пикс.

– цикл перебора значений угла
  с заданным шагом
– вычисление значения функции
– пересчет в координаты x,y
– первую точку – ставим (moveTo),
  до остальных рисуем линию
```

9. Запустим анимацию. Проверим результат:

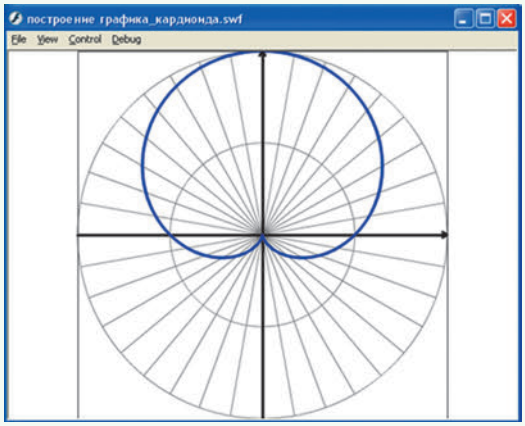


10. Задания для самостоятельного выполнения:

- “улитка Паскаля”: $R(\alpha) = 2 - 4\sin(\alpha)$:




- “кардиоида”: $R(\alpha) = 1 - \sin(\alpha)$:




11. Дополним анимацию возможностью обрабатывать более чем один “оборот” аргумента (т.е. реализуем произвольный диапазон изменения аргумента, например, от 0 до $4\pi$).

— Изменение в программном коде первого кадра (глобальные переменные):

```
hw = 400;                  Глобальные переменные:
d = 0.01;                  — высота/ширина "холста";
amax = 8*Math.PI;          — шаг изменения аргумента (угла)
                           — верхняя граница диапазона угла
```

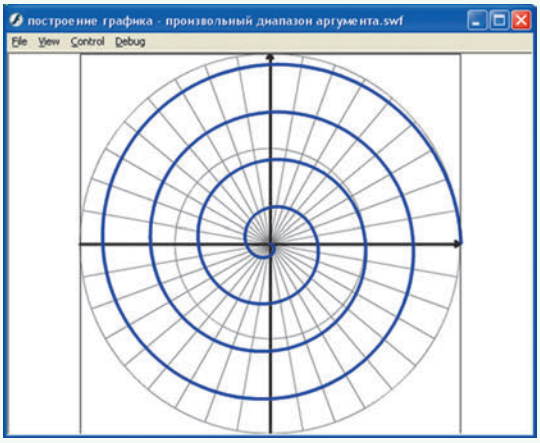
— Изменение в программном коде “холста” (фрагмент, ответственный за построение графика; в программе рисования координатной сетки ничего менять не нужно):

```
        . . .                          . . .
  this.lineStyle(3,0x0000FF,100);    — график синий толщиной 3 пикс.

  for (aa=0; aa <= _root.amax; aa+=_root.d) {   — цикл перебора значений угла
    rr = _root.f(aa);                             с заданным шагом
    x = rr*Math.cos(aa)*kR;                     — вычисление значения функции
    y = rr*Math.sin(aa)*kR;                     — пересчет в координаты x,y
    if (aa == 0) { this.moveTo(x,y); }          — первую точку — ставим (moveTo),
      else { this.lineTo(x,y); }                  до остальных рисуем линию
  }
}
```

— Функция для построения графика (“спираль Архимеда”):

```
function f(a) {              Функция для построения графика:
  r = 0.08*a;                - вычисление значения функции
  return r;                  - возврат результата
}
```

12. Запустим анимацию. Проверим результат:




### *Задание III. Построение графика функции с автоматическим масштабированием*

13. Изменим программный код, привязанный к объекту MovieClip (“холсту”):

```
onClipEvent(load) {                                  | При запуске анимации:

  this.lineStyle(1,0x888888,100);                    | – координатная сетка серая

  maxR = _root.f(0);                                 | – определение максимального
  for (aa=_root.d; aa <= _root.amax; aa+=_root.d)    |   значения радиуса
{
    if (_root.f(aa) > maxR) {
      maxR = _root.f(aa);
    }
  }

  kR = (_root.hw/2)/maxR;                            | – коэффициент масштабирования

  for (rr=1; rr <= maxR; rr++) {                     | – цикл рисования окружностей
    for (aa=0; aa <= 2*Math.PI; aa+=0.01) {          |   – вложенный цикл по углам
      x = rr*Math.cos(aa)*kR;                        | – координата x
      y = -rr*Math.sin(aa)*kR;                       | – координата y
      if (aa == 0) { this.moveTo(x,y); }             | – первая точка ставится,
        else { this.lineTo(x,y); }                   |   остальные – рисуются отрезки
    }
  }

  for (aa=0; aa <= 2*Math.PI; aa+=10*Math.PI/180)    | – цикл рисования
{                                                    |   радиусов с шагом 10 градусов
    x = maxR*Math.cos(aa)*kR;                        | – координата x
    y = -maxR*Math.sin(aa)*kR;                       | – координата y
    this.moveTo(0,0);                                | – рисовать от нулевой точки
    this.lineTo(x,y);                                |   до вычисленной на max радиусе
  }

  this.lineStyle(3,0x000000,100);                    | – оси черные толщиной 3 пикс.

  this.moveTo(-_root.hw/2,0);                        | – ось X
  this.lineTo(_root.hw/2,0);
  this.lineTo(_root.hw/2-5,-3);                      | – стрелка оси X
  this.moveTo(_root.hw/2,0);
  this.lineTo(_root.hw/2-5,3);

  this.moveTo(0,_root.hw/2);                         | – ось Y
  this.lineTo(0,-_root.hw/2);
  this.lineTo(-3,-_root.hw/2+5);                     | – стрелка оси Y
  this.moveTo(0,-_root.hw/2);
  this.lineTo(3,-_root.hw/2+5);
```

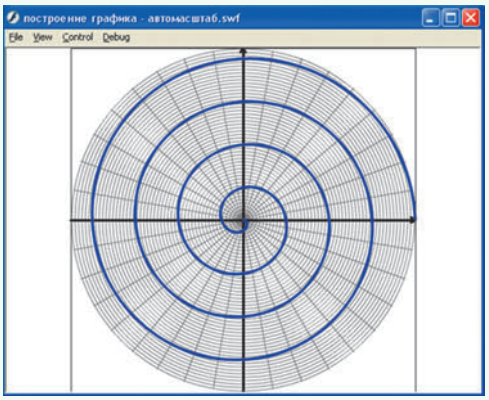
```
  this.lineStyle(3,0x0000FF,100);
  for (aa=0; aa <= 2*Math.PI;
       aa+=_root.d) {
    rr = _root.f(aa);
    x = rr*Math.cos(aa)*kR;
    y = rr*Math.sin(aa)*kR;
    if (aa == 0) { this.moveTo(x,y); }
      else { this.lineTo(x,y); }
  }
}
```

```
— график синий толщиной 3 пикс.
— цикл перебора значений угла
  с заданным шагом
— вычисление значения функции
— пересчет в координаты x,y

— первую точку — ставим (moveTo),
  до остальных рисуем линию
```

14. Зададим для построения функцию

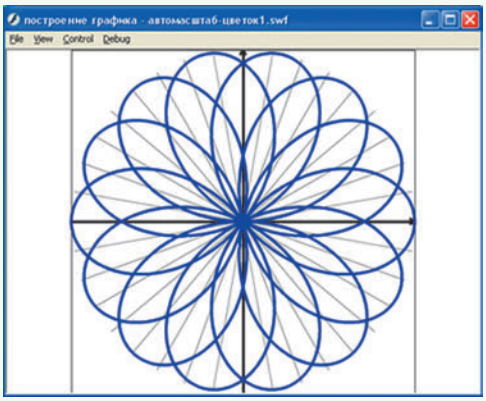
```
function f(a) {
  r = 2*a;
  return r;
}
```

Запустим анимацию. Получим результат:

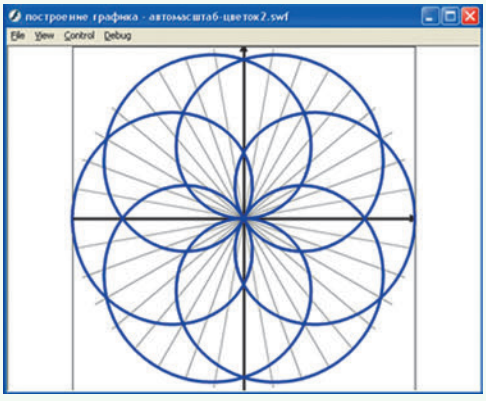



15. Задания для самостоятельного выполнения:

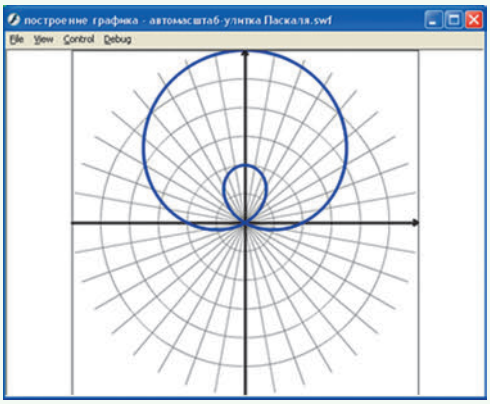
- $R(\alpha)=\sin\left(\frac{7}{4}\alpha\right),\ \alpha\in[0,8\pi]$:




- $R(\alpha)=\sin\left(\frac{3}{4}\alpha\right),\ \alpha\in[0,8\pi]$:



- “улитка Паскаля”: $R(\alpha) = 2 - 4\sin(\alpha)$:




## УРОК 14. ФИГУРЫ ЛИССАЖУ

Фигуры Лиссажу — замкнутые траектории, прочерчиваемые точкой, совершающей одновременно два гармонических колебания в двух взаимно перпендикулярных направлениях. Впервые изучены французским ученым Жюлем Антуаном Лиссажу. Вид фигур зависит от соотношения между периодами (частотами), фазами и амплитудами обоих колебаний.

(“Википедия”: *https://ru.wikipedia.org/?oldid=61369347*)

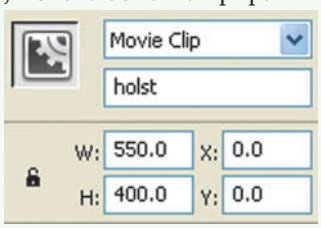
Принцип сложения гармонических колебаний по осям *X* и *Y*, приводящего к получению фигуры Лиссажу.

## Практические задания

### *Задание I. Рисование системы координат*

1. Создадим новую анимацию. Будем рисовать в объекте MovieClip с выводом графика сразу после запуска анимации.

2. Нарисуем в первом кадре прямоугольник с выключенным фоном. Размеры прямоугольника зададим так, чтобы он занимал максимум площади кадра анимации. Установим его размер равным 550 × 400 (размеры кадра анимации), а координаты *X*, *Y* этого объекта приравняем нулю.




Преобразуем прямоугольник в символ MovieClip (“холст”). Точку привязки MovieClip разместим в центре.

3. Присвоим первому кадру программный код:

```
h0 = 400;
w0 = 550;
AX = 2;
AY = 2;
```

```
Глобальные переменные:
– высота "холста";
– ширина "холста";
– амплитуда по оси X
– амплитуда по оси Y
```

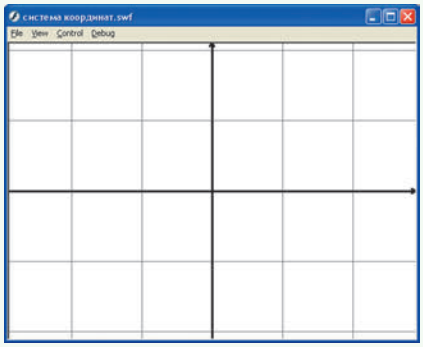
4. Присвоим объекту MovieClip (“холсту”) программный код для события **load** (с некоторыми изменениями — автоматическое масштабирование по значениям амплитуд по осям *X* и *Y*; расчет координат от центра “холста”):

```
onClipEvent(load) {
  kx = (_root.w0/2-10)/_root.AX;
  ky = (_root.h0/2-10)/_root.AY;
  kx = Math.min(kx,ky); ky = kx;

  this.lineStyle(1,0x888888,100);
  for (xx = -_root.AX; xx <= _root.AX; xx++) {
    this.moveTo(xx*kx,-_root.h0/2);
    this.lineTo(xx*kx,_root.h0/2);
  }

  for (yy = -_root.AY; yy <= _root.AY; yy++) {
    this.moveTo(-_root.w0/2,yy*ky);
    this.lineTo(_root.w0/2,yy*ky);
  }

  this.lineStyle(3,0x000000,100);
  this.moveTo(-_root.w0/2,0);
  this.lineTo(_root.w0/2,0);
  this.lineTo(_root.w0/2-5,-3);
  this.moveTo(_root.w0/2,0);
  this.lineTo(_root.w0/2-5,3);

  this.moveTo(0,_root.h0/2);
  this.lineTo(0,-_root.h0/2);
  this.lineTo(-3,-_root.h0/2+5);
  this.moveTo(0,-_root.h0/2);
  this.lineTo(3,-_root.h0/2+5);

}
```

```
При запуске анимации:
– коэффициент масштаба по оси X
– коэффициент масштаба по оси Y
– выбираем коэффициенты равными
  наименьшему из них
– координатная сетка серая
– цикл рисования вертикальных
  линий координатной сетки

– цикл рисования горизонтальных
  линий координатной сетки

– оси черные толщиной 3 пикс.
– ось X

– стрелка оси X

– ось Y

– стрелка оси Y
```

При расчетах коэффициентов масштабирования дополнительно учитывается отступ 10 пикселей от каждого края, чтобы линия графика не прилегала к краям. Далее выбирается наименьшее значение из двух полученных и оно берется как значение обоих коэффициентов. Это важно, так как нужно сохранить пропорции построения по оси *X* и *Y* равными.

5. Запустим анимацию. Результат должен выглядеть так:




### *Задание II. Построение фигуры Лиссажу для синхронных колебаний*

Общие параметрические уравнения фигур Лиссажу (параметр — время *t*):

$$\begin{cases} x = AX \cdot \sin(fx \cdot t + d), \\ y = AY \cdot \sin(fy \cdot t) \end{cases}$$

где *AX*, *AY* — амплитуды колебаний по осям *X* и *Y*, *fx* и *fy* — частоты колебаний по осям *X* и *Y*, *d* — сдвиг фаз колебаний по осям *X* и *Y*

Построим график фигуры Лиссажу, соответствующей равным значениям амплитуд ($AX = AY$) и частот ($a = b$) и значению сдвига фаз, равному $\pi/2$.

6. Откроем ранее созданную анимацию “построение системы координат” и сохраним под новым именем (“построение фигуры Лиссажу 1”).

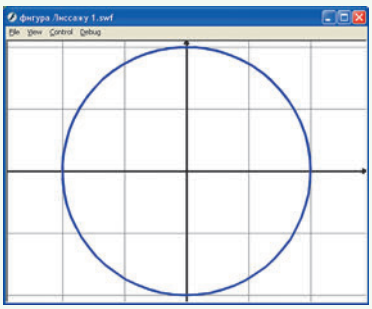
7. Программный код, привязанный к первому кадру анимации, дополним глобальными переменными, определяющими значения параметров колебаний по осям *X* и *Y*:

```
h0 = 400;                  Глобальные переменные:
w0 = 550;                  — высота "холста";
AX = 2;                    — ширина "холста";
AY = 2;                    — амплитуда по оси X
fx = 10;                   — амплитуда по оси Y
fy = 10;                   — частота по оси X
d = Math.PI/2;             — частота по оси Y
t0 = 0;                    — сдвиг фаз
tk = 1;                    — интервал изменения времени
dt = 0.01;                   [0,1] (секунд)
                           — шаг изменения t
```

8. Программный код, привязанный к объекту MovieClip (“холсту”), дополним построением фигуры Лиссажу:

```
onClipEvent(load) {                                  При запуске анимации:
  kx = (_root.w0/2-10)/_root.AX;                     — коэффициент масштаба по оси X
  ky = (_root.h0/2-10)/_root.AY;                     — коэффициент масштаба по оси Y
  kx = Math.min(kx,ky); ky = kx;                     — выбираем коэффициенты равными
                                                       наименьшему из них
  this.lineStyle(1,0x888888,100);                    — координатная сетка серая
  for (xx = -_root.AX; xx <= _root.AX; xx++) {       — цикл рисования вертикальных
    this.moveTo(xx*kx,-_root.h0/2);                    линий координатной сетки
    this.lineTo(xx*kx,_root.h0/2);
  }

  for (yy = -_root.AY; yy <= _root.AY; yy++) {       — цикл рисования горизонтальных
    this.moveTo(-_root.w0/2,yy*ky);                    линий координатной сетки
    this.lineTo(_root.w0/2,yy*ky);
  }

  this.lineStyle(3,0x000000,100);                    — оси черные толщиной 3 пикс.
  this.moveTo(-_root.w0/2,0);                        — ось X
  this.lineTo(_root.w0/2,0);
  this.lineTo(_root.w0/2-5,-3);                      — стрелка оси X
  this.moveTo(_root.w0/2,0);
  this.lineTo(_root.w0/2-5,3);

  this.moveTo(0,_root.h0/2);                         — ось Y
  this.lineTo(0,-_root.h0/2);
  this.lineTo(-3,-_root.h0/2+5);                     — стрелка оси Y
  this.moveTo(0,-_root.h0/2);
  this.lineTo(3,-_root.h0/2+5);

  this.lineStyle(3,0x0000FF,100);                    — линии фигуры Лиссажу — синие
  for (t = _root.t0; t <= _root.tk; t += _root.      — цикл изменения параметра времени t
dt) {
    x = _root.AX * Math.sin(_root.fx*t+_root.d);     — вычисление координаты x
    y = _root.AY*Math.sin(_root.fy*t);               — вычисление координаты y
    x = Math.round(x*kx);                            — пересчет x в масштабе
    y = -Math.round(y*ky);                           — пересчет y в масштабе
    if (t==_root.t0) { this.moveTo(x,y); }           — первую точку ставим,
      else { this.lineTo(x,y); }                       все другие — рисуем отрезки
  }

}
```

9. Запустим анимацию. Проверим результат:

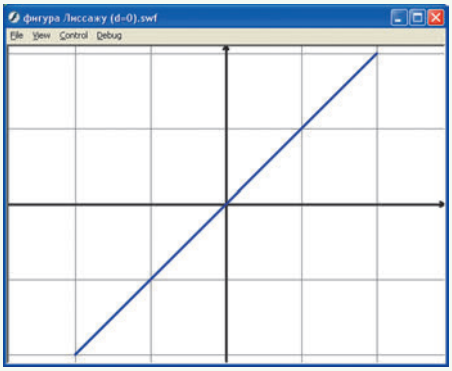


При указанных условиях (равные амплитуды, равные частоты, сдвиг фаз в $\pi/2$) фигура Лиссажу соответствует окружности.

***Задание III. Исследование фигур Лиссажу***

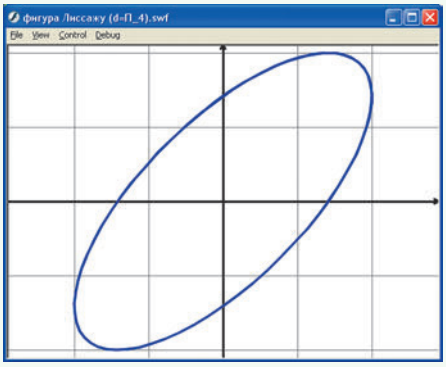
Исследуем зависимость вида фигуры Лиссажу от сдвига фаз.

10. Установим (в списке глобальных переменных в скриптовом коде, привязанном к кадру) значение *d* равным нулю. Запустим анимацию. Получаемый результат:



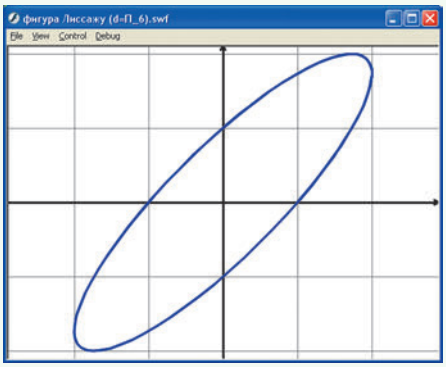

При заданных условиях (равные амплитуды и частоты, сдвиг фаз равен нулю) фигура Лиссажу — это прямая, наклоненная к оси *X* под углом 45 градусов.

11. Установим (в списке глобальных переменных в скриптовом коде, привязанном к кадру) значение *d* равным любому значению между 0 и $\pi/2$, например, $\pi/4$ (**Math.PI/4**). Запустим анимацию. Получаемый результат:



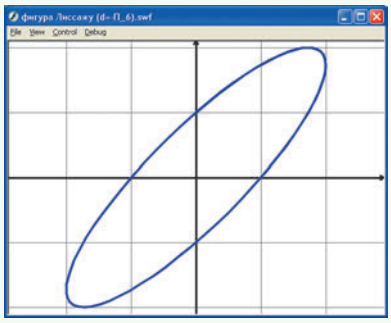

При заданных условиях (равные амплитуды и частоты, сдвиг фаз лежит в интервале от 0 до $\pi/2$) фигура Лиссажу — эллипс, большая ось которого наклонена к оси *X* под углом 45 градусов.

12. Установим значение *d* равным $\pi/6$ (**Math.PI/6**). Запустим анимацию. Получаемый результат:



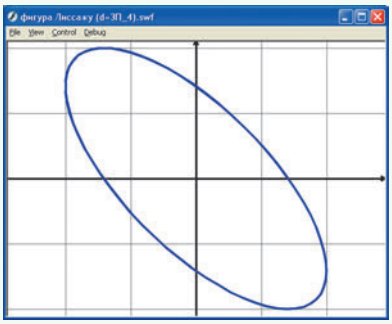

Уменьшение значения разности фаз приводит к сжатию эллипса (он “стягивается” в прямую), а увеличение значения разности фаз до $\pi/2$ приводит к расширению эллипса (он “растягивается” в окружность).

13. Установим значение *d* равным отрицательному значению $-\pi/6$ (**–Math.PI/6**). Запустим анимацию. Получаемый результат:



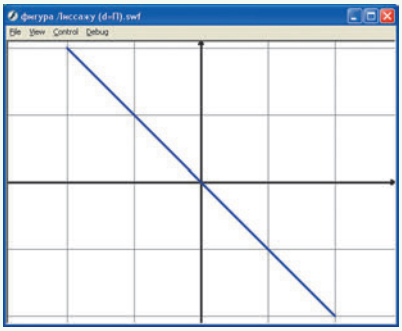
Знак разности фаз не влияет на вид фигуры Лиссажу.

14. Установим значение *d* равным любому значению между π/2 и π, например, 3π/4 (3***Math.PI/4**). Запустим анимацию. Получаемый результат:



При заданных условиях (равные амплитуды и частоты, сдвиг фаз лежит в интервале от π/2 до π) фигура Лиссажу — эллипс, большая ось которого наклонена к оси *X* под углом 135 градусов.

15. Установим значение *d* равным π. Запустим анимацию. Получаемый результат:

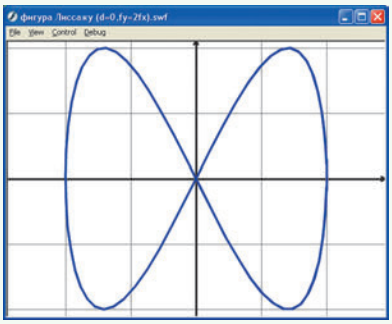


Если сдвиг фаз равен π, фигура Лиссажу — прямая, наклоненная к оси *X* под углом 135 градусов.

Можно сформулировать правило: “Большая ось эллипса (либо прямая при сдвиге фаз, равном 0 или π) направлена по середине той четверти координатной плоскости, которой соответствует значение угла разности фаз, и продолжается центрально симметрично в противолежащую четверть”.

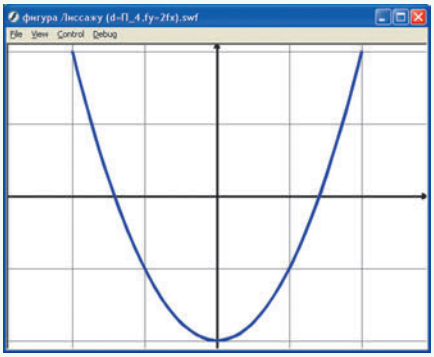
Исследуем поведение фигуры Лиссажу при изменении соотношения частот.

16. Установим значение разности фаз равным 0, а частоту *fy* вдвое большей *fx*. Результат:



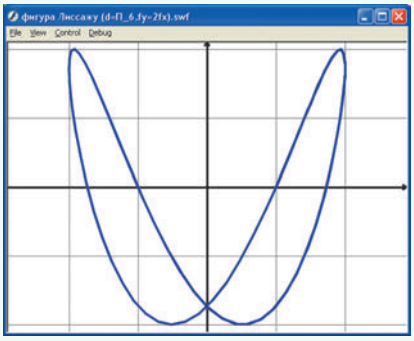
Фигура Лиссажу имеет вид “восьмерки”, так как за то же время, пока по оси *X* совершается одно колебание, по оси *Y* совершается два колебания.

17. Установим значение разности фаз равным π/4, а частоту *fy* вдвое большей *fx*. Результат:



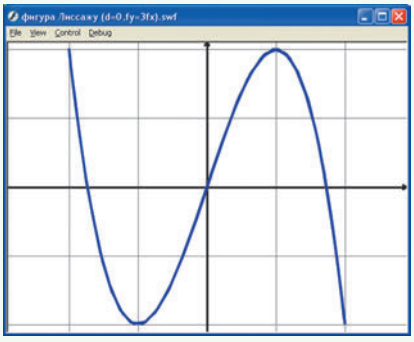

Фигура “вырождается” в параболу.

18. Установим значение разности фаз равным π/6, а частоту *fy* вдвое большей *fx*. Результат:



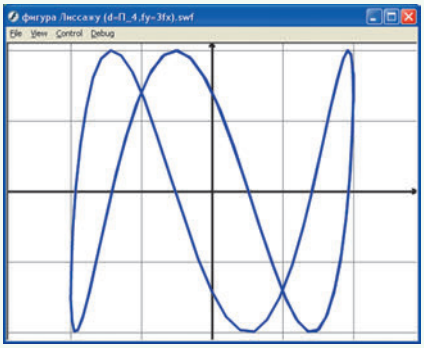

Меняя значение разности фаз, можно проследить, как “восьмерка” раскрывается из параболы, а затем снова “схлопывается” в параболу. (Исследуйте этот процесс для изменения разности фаз в интервале [0, π] с шагом π/6. Расположите получаемые скриншоты в ряд с указанием значений разности фаз.)

19. Установим значение разности фаз равным 0, а частоту *fy* втрое большей *fx*. Результат:



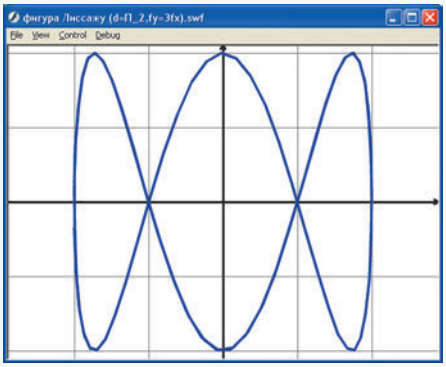

Фигура приобрела вид кривой, как бы “слитой” из двух парабол.

20. Установим значение разности фаз равным π/4, а частоту *fy* втрое большей *fx*. Результат:



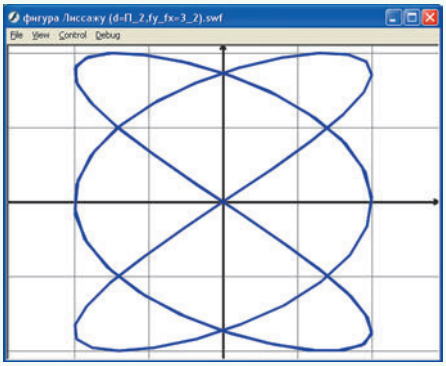

Кривая “раскрывается” в петлю.

21. Установим значение разности фаз равным π/2, а частоту *fy* втрое большей *fx*. Результат:

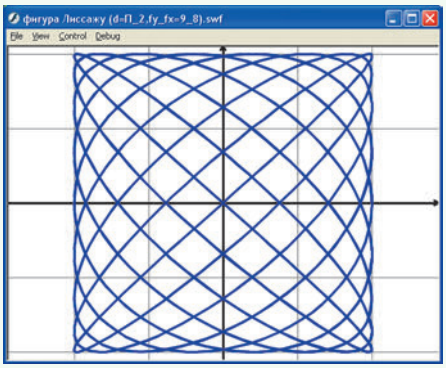



Кривая полностью раскрыта.

22. Установим значение разности фаз равным π/2, а соотношение частот *fx*/*fy* равным 3/2. Результат:

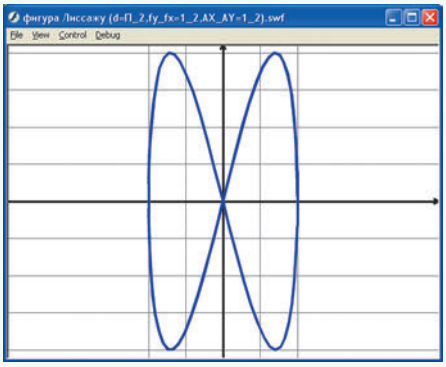



23. Установим значение разности фаз равным π/2, а соотношение частот *fx*/*fy* равным 9/8. Для большей гладкости построения линии выберем частоты равными, соответственно, 9 и 8, а конечное значение t равным 10. Результат:




Исследуем поведение фигур Лиссажу при изменении соотношения частот.

24. Установим значение разности фаз равным π/2, а соотношение частот *fx*/*fy* равным 1/2. (Выберем частоты равными, соответственно, 10 и 20, а конечное значение t равным 1.) Установим значение амплитуды *AY* вдвое большим, чем *AX*. Результат:




Изменение соотношения амплитуд приводит к соответствующему сжатию фигуры вдоль оси, соответствующей меньшей амплитуде.

### *Задание IV. Построение семейства фигур Лиссажу*

Выполним построение набора фигур Лиссажу для разных значений сдвига фаз от 0 до $\pi/2$ с шагом $\pi/8$ при соотношении частот *fx/fy* равным 1 и равных амплитудах.

Реализуем изменение яркости цвета строящихся кривых для разных значений сдвига фаз. Для этого используем следующую операцию, которая для значений цветовых компонентов R, G, B (от 0 до 255) возвращает код цвета: **(*R*<<16|*G*<<8|*B*)**.

Можно использовать и более простой вариант — менять значение константы цвета от 0 до шестнадцатеричного значения FFFF88, или десятичного 16777096 (значение FFFFFF — белый цвет, будет не виден на белом фоне), с шагом, вычисляемым исходя из вычисленного количества шагов изменения значения сдвига фаз.

25. Программный код, привязанный к первому кадру анимации, дополним глобальными переменными, определяющими изменение значений сдвига фаз и изменение значений цвета:

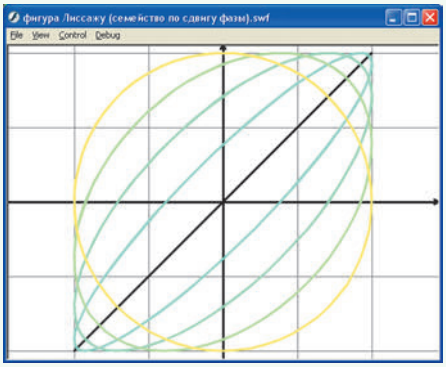
```
h0 = 400;          // Глобальные переменные:
w0 = 550;          // – высота "холста";
AX = 2;            // – ширина "холста";
AY = 2;            // – амплитуда по оси X
fx = 10;           // – амплитуда по оси Y
fy = 10;           // – частота по оси X
d = Math.PI/2;     // – частота по оси Y
t0 = 0;            // – сдвиг фаз
tk = 1;            // – интервал изменения времени
dt = 0.01;         //   [0,1] (секунд)
d0 = 0;            // – шаг изменения t
dk = Math.PI/2;    // – интервал изменения сдвига фаз
dd = Math.PI/8;    //   равен [0,π]
                   // – шаг изменения сдвига фаз
```

26. Дополним программный код, привязанный к объекту MovieClip (“холсту”):

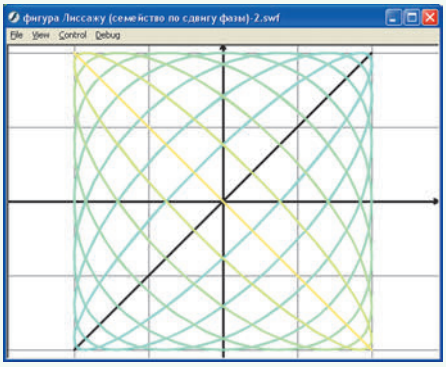
```
onClipEvent(load) {
  kx = (_root.w0/2-10)/_root.AX;
  ky = (_root.h0/2-10)/_root.AY;
  kx = Math.min(kx,ky); ky = kx;

  this.lineStyle(1,0x888888,100);
  for (xx = -_root.AX;
       xx <= _root.AX; xx++) {
    this.moveTo(xx*kx,-_root.h0/2);
    this.lineTo(xx*kx,_root.h0/2);
  }
  for (yy = -_root.AY;
       yy <= _root.AY; yy++) {
    this.moveTo(-_root.w0/2,yy*ky);
    this.lineTo(_root.w0/2,yy*ky);
  }
  this.lineStyle(3,0x000000,100);
  this.moveTo(-_root.w0/2,0);
  this.lineTo(_root.w0/2,0);
  this.lineTo(_root.w0/2-5,-3);
  this.moveTo(_root.w0/2,0);
  this.lineTo(_root.w0/2-5,3);
  this.moveTo(0,_root.h0/2);
  this.lineTo(0,-_root.h0/2);
  this.lineTo(-3,-_root.h0/2+5);
  this.moveTo(0,-_root.h0/2);
  this.lineTo(3,-_root.h0/2+5);
  col = 0;
  for (ddd = _root.d0; ddd <= _root.dk;
       ddd += _root.dd) {
    this.lineStyle(3,col,100);
    for (t = _root.t0; t <= _root.tk; t += _root.dt) {
      x = _root.AX * Math.sin(_root.fx*t+ddd);

      y = _root.AY*Math.sin(_root.fy*t);
      x = Math.round(x*kx);
      y = -Math.round(y*ky);
      if (t==_root.t0) { this.moveTo(x,y); }
        else { this.lineTo(x,y); }
    }
  col += 16777096/Math.round((_root.dk-_root.d0)/_root.dd);
  }
}
```

```
При запуске анимации:
– коэффициент масштаба по оси X
– коэффициент масштаба по оси Y
– выбираем коэффициенты равными
  наименьшему из них
– координатная сетка серая
– цикл рисования вертикальных
  линий координатной сетки


– цикл рисования горизонтальных
  линий координатной сетки


– оси черные толщиной 3 пикс.
– ось X

– стрелка оси X


– ось Y

– стрелка оси Y


– изначально цвет черный
– цикл изменения значений
  сдвига фаз
– линии текущего цвета
– цикл изменения параметра времени t
– вычисление координаты x
  при текущем сдвиге фаз
– вычисление координаты y
– пересчет x в масштабе
– пересчет y в масштабе
– первую точку ставим,
  все другие – рисуем отрезки

– вычисляем новое значение цвета
```

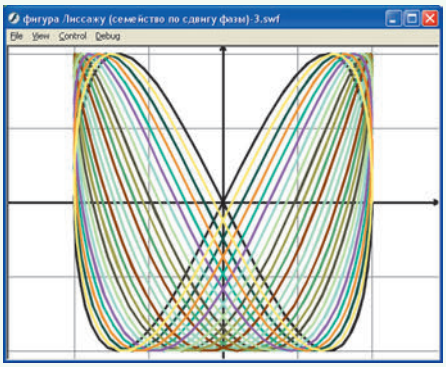
27. Запустим анимацию. Получим результат, демонстрирующий изменение вида фигур Лиссажу при изменении сдвига фаз:




28. Построим семейство фигур Лиссажу при изменении сдвига фаз в диапазоне от 0 до π:




29. Выполним построение набора фигур Лиссажу для разных значений сдвига фаз от 0 до π/2 с шагом π/25 при соотношении частот *fx/fy* равным 1/2 и равных амплитудах:




Самостоятельно реализуйте построение семейств фигур Лиссажу при постоянных значениях сдвига фаз (значения 0, π/6, π/4, π/3, π/2) и изменении соотношения частот *fx/fy* от 0 до 8 с шагом 0,25.

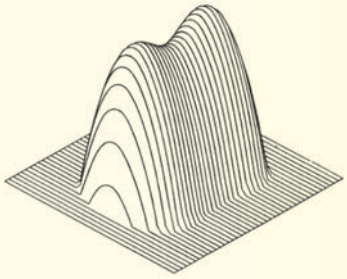
## УРОК 15. ПОСТРОЕНИЕ ТРЕХМЕРНЫХ ГРАФИКОВ ФУНКЦИЙ ОТ ДВУХ ПЕРЕМЕННЫХ

### Принципы построения трехмерных изображений

Трехмерное изображение представляет собой виртуальную модель, состоящую из точек и отрезков, где точки (концы отрезков) имеют три координаты — *x*, *y*, *z*.

При построении трехмерного графика от двух переменных координаты *x* и *y* определяются значениями исходных переменных, а третья координата точки графика *z* вычисляется как значение функции *F*(*x*, *y*). Такой график представляет собой поверхность, которую можно изобразить как набор двумерных графиков, каждый из которых, например, вычерчивается для одного из значений *y* при изменении значения *x*. Таким образом изображается как бы “набор срезов” трехмерной поверхности.

Пример трехмерного графика, построенного таким способом:

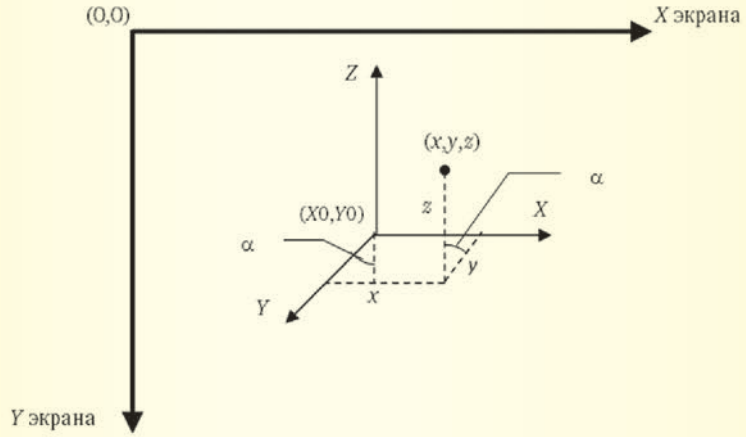


На экране дисплея фактически строится проекция рассчитываемого трехмерного изображения на плоскость экрана.

Такая проекция может быть получена различными способами.

1. **Диметрическая проекция:** ось *x* направлена горизонтально вправо, ось *y* направлена под углом 45 градусов влево вниз, ось *z* направлена вертикально вверх. Такой способ проекции упрощает расчет координат при построении изображения, но приводит к искажениям изображаемого 3D-объекта.

Пересчет исходных трех координат (*x*, *y*, *z*) точек в экранные координаты (*x*1, *y*1) можно произвести по следующим формулам (см. чертеж).

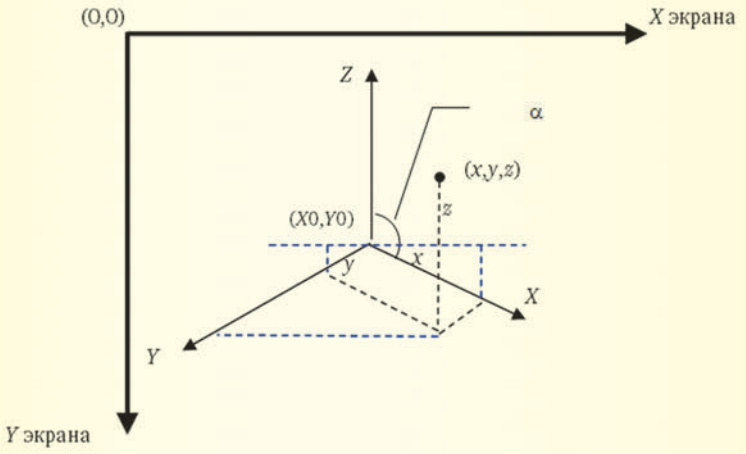



$$xkomp = X0 + kx*x - ky*y*\sin(\alpha);$$
$$ykomp = Y0 + ky*y*\cos(\alpha) - kz*z$$

где *kx*, *ky*, *kz* — коэффициенты масштабирования по осям.

2. **Аксонометрическая проекция:** ось *z* направлена вертикально вверх, а оси *x* и *y* направлены под наклоном вниз вправо и вниз влево соответственно, так что все углы между осями составляют 120 градусов.

Пересчет исходных трех координат (*x*,*y*,*z*) точек в экранные координаты (*x*1,*y*1) можно произвести по следующим формулам (см. чертеж).



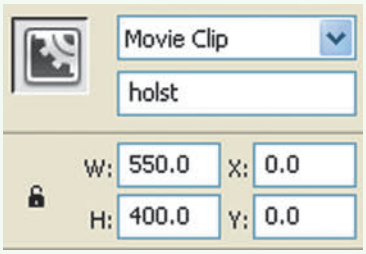

$$xkomp = X0 + kx*x*\cos(\alpha - 90) - ky*y*\cos(\alpha - 90);$$
$$ykomp = Y0 + kx*x*\sin(\alpha - 90) + ky*y*\sin(\alpha) - kz*z$$

где *kx*, *ky*, *kz* — коэффициенты масштабирования по осям.

## Практические задания

### *Задание I. Рисование системы координат*

1. Создадим новую анимацию с размером кадра 550 × 400. Нарисуем в первом кадре прямоугольник с выключенным фоном. Размеры прямоугольника зададим так, чтобы он занимал максимум площади кадра анимации. Установим его размер равным 550 × 400 (размеры кадра анимации), а координаты *X*,*Y* этого объекта приравняем нулю.




Преобразуем прямоугольник в символ MovieClip (“холст”). Точку привязки MovieClip разместим в верхнем левом углу. Даем объекту имя экземпляра **xolst1**.

2. Присвоим первому кадру программный код:

```
h0 = 400;
w0 = 550;
X0 = w0/2;
Y0 = h0/2;
AL = 45;
ALR = AL* Math.PI/180;
XMAX = 10;
YMAX = 10;
ZMAX = 10;
KZ = (Y0-10)/ZMAX;
KX = (w0-X0-10)/XMAX;
KY = Math.min(X0-10, h0-Y0-10)/Math.
cos(ALR)/YMAX;
```

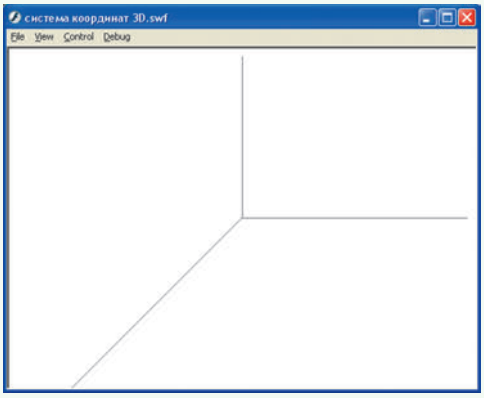
```
Глобальные переменные:
– высота "холста";
– ширина "холста";
– координата X 3D системы координат
– координата Y 3D системы координат
– угол наклона оси y в градусах
– угол наклона оси y в радианах
– максимальное значение координаты x
– максимальное значение координаты y
– максимальное значение координаты z
– коэффициент масштаба по z
– коэффициент масштаба по x
– коэффициент масштаба по y
```

3. Присвоим объекту MovieClip (“холсту”) программный код для события **load**:

```
onClipEvent(load) {
  this.lineStyle(1,0x888888,100);

  this.moveTo(_root.X0,_root.Y0);
  this.lineTo(_root.X0,_root.Y0-_root.KZ*_root.ZMAX);

  this.moveTo(_root.X0,_root.Y0);
  this.lineTo(_root.X0+_root.KX*_root.XMAX,
              _root.Y0);

  this.moveTo(_root.X0,_root.Y0);
  this.lineTo(_root.X0-_root.KY*_root.YMAX*
              Math.sin(_root.ALR),_root.Y0+
              _root.KY*_root.YMAX*
              Math.cos(_root.ALR));
}
```

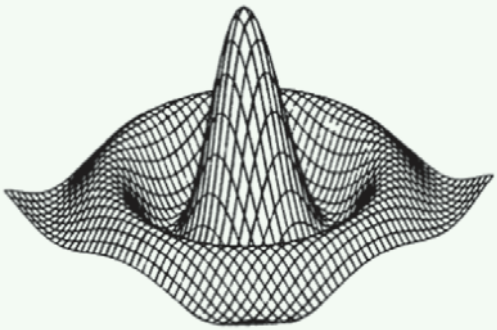
```
При запуске анимации:
– оси – серый цвет

– рисовать от точки (X0,Y0) ось Z


– рисовать от точки (X0,Y0) ось X


– рисовать от точки (X0,Y0) ось Y
```

4. Запустим анимацию. Результат должен выглядеть так:



### *Задание II. Построение 3D-графика в диметрической проекции*

Построим график функции $z = \frac{\sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}$.



5. Откроем ранее созданную анимацию “построение системы координат” и сохраним под новым именем (“построение 3D-графика”).

6. Дополним программный код, привязанный к первому кадру анимации:

```
h0 = 400;
w0 = 550;
X0 = w0/2;
Y0 = h0/2;
AL = 45;
ALR = AL* Math.PI/180;
XMAX = 17;
YMAX = 17;
ZMAX = 1;
KZ = (Y0-10)/ZMAX;
KX = (w0-X0-10)/XMAX;
KY = Math.min(X0-10, h0-Y0-10)/Math.
cos(ALR)/YMAX;

X1 = -10;
X2 = 10;
DX = (X2-X1)/100;

Y1 = -10;
Y2 = 10;
DY = (Y2-Y1)/100;

function f(x,y) {
  r = Math.sqrt(x*x+y*y);
  if (r == 0) { r=0.001 }
  z = Math.sin(r)/r;
  return z;
}
```

```
Глобальные переменные:
– высота "холста";
– ширина "холста";
– координата X 3D системы координат
– координата Y 3D системы координат
– угол наклона оси y в градусах
– угол наклона оси y в радианах
– максимальное значение координаты x
– максимальное значение координаты y
– максимальное значение координаты z
– коэффициент масштаба по z
– коэффициент масштаба по x
– коэффициент масштаба по y


– диапазон изменения значений x
  (от X1 до X2)
– шаг изменения X

– диапазон изменения значений y
  (от Y1 до Y2)
– шаг изменения Y

Функция для построения графика:

– исключить деление на ноль
– вычисление значения функции
– возврат результата
```

Значения XMAX, YMAX и ZMAX подбираются вручную, чтобы график полностью располагался внутри “холста” и занимал максимум места на нем.

7. Программный код, привязанный к объекту MovieClip (“холсту”), дополним построением 3D-графика:

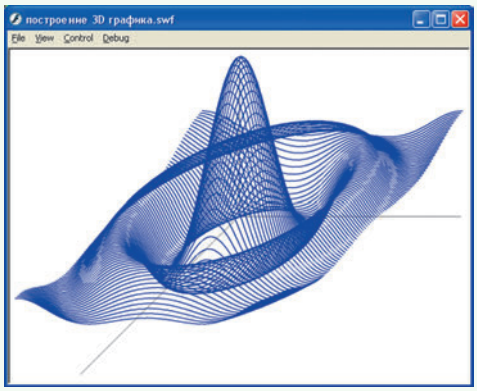
```
onClipEvent(load) {

  this.lineStyle(1,0x888888,100);

  this.moveTo(_root.X0,_root.Y0);
  this.lineTo(_root.X0,_root.Y0-_root.KZ*_
root.ZMAX);

  this.moveTo(_root.X0,_root.Y0);
  this.lineTo(_root.X0+_root.KX*_root.XMAX,
              _root.Y0);
```

```
При запуске анимации:

– оси – серый цвет

– рисовать от точки (X0,Y0) ось Z


– рисовать от точки (X0,Y0) ось X
```

```
  this.moveTo(_root.X0,_root.Y0);
  this.lineTo(_root.X0-_root.KY*_root.YMAX*
       Math.cos(_root.ALR),_root.Y0 +_root.KY*
                _root.YMAX*Math.sin(_root.ALR));
  this.lineStyle(1,0x0000FF,100);
  for (yy=_root.Y1; yy<=_root.Y2; yy+=_root.DY) {

    for (xx=_root.X1; xx<=_root.X2; xx+=_root.DX)
{
       zz = _root.f(xx,yy);

       x = _root.X0 + _root.KX*xx -
           _root.KY*yy*Math.sin(_root.ALR);
       y = _root.Y0 + _root.KY*yy*
           Math.cos(_root.ALR) - _root.KZ*zz;
       if (xx==_root.X1) { this.moveTo(x,y); }
           else { this.lineTo(x,y); }
   }
  }
}
```

```
- рисовать от точки (X0,Y0)
  ось Y


- график синий толщиной 1 пикс.
- внешний цикл - перебор
  значений координаты y
- внешний цикл - перебор
  значений координаты x
- вычисление значения функции

- пересчет трехмерных координат
  в экранные


- первую точку - ставим,
  до остальных рисуем линию
```

8. Запустим анимацию. Проверим результат:




### *Задание III. Построение 3D-графика в аксонометрической проекции*

9. Откроем предыдущую анимацию и сохраним под новым именем (“построение 3D-графика 2”).
10. Изменим программный код, привязанный к первому кадру анимации:

```
h0 = 400;
w0 = 550;
X0 = w0/2;
Y0 = h0/2;
AL = 120;
ALR = (AL-90)* Math.PI/180;
XMAX = 21;
YMAX = 21;
ZMAX = 1;
KZ = (Y0-10)/ZMAX;
KX = Math.sqrt((w0-X0-10)*
     (w0-X0-10)+(h0-Y0-10)*
     (h0-Y0-10))/XMAX;
KY = KX;
X1 = -10;
X2 = 10;
DX = (X2-X1)/100;
Y1 = -10;
Y2 = 10;
DY = (Y2-Y1)/100;
function f(x,y) {
  r = Math.sqrt(x*x+y*y);
  if (r == 0) { r=0.001 }
  z = Math.sin(r)/r;
  return z;
}
```

```
Глобальные переменные:
- высота "холста";
- ширина "холста";
- координата X 3D системы координат
- координата Y 3D системы координат
- угол наклона осей x и y в градусах
- угол наклона осей x и y в радианах
- максимальное значение координаты x
- максимальное значение координаты y
- максимальное значение координаты z
- коэффициент масштаба по z
- коэффициент масштаба по x


- коэффициент масштаба по y (такой же)
- диапазон изменения значений x
  (от X1 до X2)
- шаг изменения X
- диапазон изменения значений y
  (от Y1 до Y2)
- шаг изменения Y
Функция для построения графика:

- исключить деление на ноль
- вычисление значения функции
- возврат результата
```

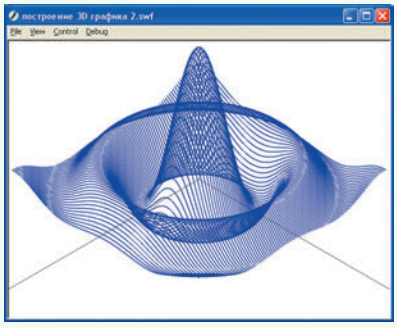
Значения XMAX, YMAX и ZMAX подбираются вручную, чтобы график полностью располагался внутри “холста” и занимал максимум места на нем.

11. Изменим программный код, привязанный к объекту MovieClip (“холсту”):

```
onClipEvent(load) {

  this.lineStyle(1,0x888888,100);

  this.moveTo(_root.X0,_root.Y0);
  this.lineTo(_root.X0,_root.Y0-
              _root.KZ*_root.ZMAX);
  this.moveTo(_root.X0,_root.Y0);
  this.lineTo(_root.X0+_root.KX*
      _root.XMAX*Math.cos(_root.ALR),
      _root.Y0+_root.KX*_root.XMAX
      *Math.sin(_root.ALR));

  this.moveTo(_root.X0,_root.Y0);
  this.lineTo(_root.X0-_root.KY*
      _root.YMAX*Math.cos(_root.ALR),
      _root.Y0+_root.KY*_root.YMAX*
      Math.sin(_root.ALR));

  this.lineStyle(1,0x0000FF,100);
  for (yy=_root.Y1; yy<=_root.Y2; yy+=_root.DY) {

    for (xx=_root.X1; xx<=_root.X2; xx+=_root.DX) {

      zz = _root.f(xx,yy);

      x = _root.X0 + _root.KX*xx*
          Math.cos(_root.ALR) - _root.KY*yy*
          Math.cos(_root.ALR);
      y = _root.Y0 + _root.KX*xx*
          Math.sin(_root.ALR) + _root.KY*yy*
          Math.sin(_root.ALR) - _root.KZ*zz;

      if (xx == _root.X1) { this.moveTo(x,y); }
           else { this.lineTo(x,y); }
    }
  }
}
```

```
При запуске анимации:

- оси - серый цвет

- рисовать от точки (X0,Y0)
  ось Z
- рисовать от точки (X0,Y0)
  ось X

- рисовать от точки (X0,Y0)
  ось Y

- график синий толщиной 1 пикс.
- внешний цикл - перебор
  значений координаты y
- внешний цикл - перебор
  значений координаты x
- вычисление значения функции

- пересчет трехмерных координат в
экранные

- первую точку - ставим,
  до остальных рисуем линию
```

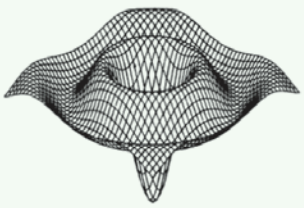
12. Запустим анимацию. Проверим результат:



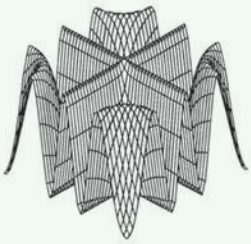
Для самостоятельного выполнения:

Постройте в аксонометрии графики функций:

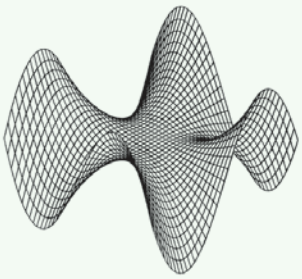
1) $z = -\frac{\sin(\sqrt{x^2+y^2})}{\sqrt{x^2+y^2}}$.



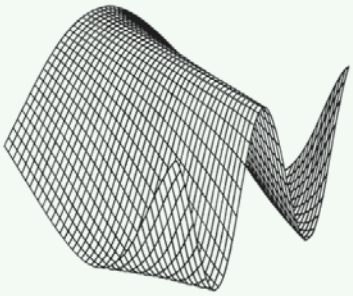
2) $z = \cos(x, y)$.



3) $z = \dfrac{x \cdot y \cdot (x-y) \cdot (x+y)}{\sqrt{x^2+y^2}}$ (“обезьянье седло”).



4) $z = \cos(x \cdot e^{-\frac{y}{5}})$.



### *Задание IV. Построение стереоскопического изображения 3D-графика в аксонометрической проекции*

Стереоскопическое (объемное) изображение представляет собой два таких изображения объекта, которые получаются при его рассматривании с двух точек, расположенных на одной высоте и разнесенных по горизонтали на некоторое расстояние. Фактически это соответствует рассматриванию объекта, соответственно, левым и правым глазами. Расстояние между точками, с которых рассматривается объект, называется стереобазисом и обычно равно расстоянию между глазами человека (примерно 65 мм).

Полученная пара изображений называется стереопарой. Для получения эффекта объема требуется, чтобы левый глаз видел только левый кадр стереопары, а правый глаз видел только правый кадр стереопары. Это реализуется различными способами, простейший из которых — рассматривание перекрестной стереопары (в которой левый кадр расположен справа, а правый кадр — слева).

Подробнее о стереоскопии, получении и рассматривании стереоскопических изображений см. в статьях:

— *Усенков Д.Ю.* Стереотехнологии в школе: как и для чего // Информатика, 2012, № 9. С. 18–26.

— *Усенков Д.Ю.* YouTube: теперь — 3D! // Информатика, 2013, № 1. С. 28–39.

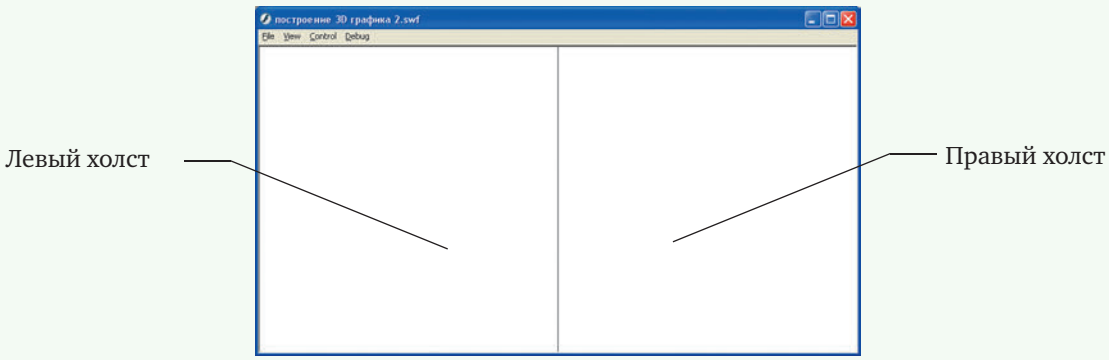
— *Усенков Д.Ю.* 3D-поверхности в Excel // Информатика, 2013, № 1. С. 40–45.

— *Богомолова О.Б., Усенков Д.Ю.* Занимательный практикум: изучаем 3D-технологии // Информатика, 2014, № 5. С. 48–55.

В нашем случае для получения стереопары с изображением 3D-графика мы предусмотрим в окне анимации два “холста”, на каждом из которых будем выводить одно и то же изображение 3D-графика. При этом используем более простую диметрическую проекцию и будем менять значение угла α: на левом “холсте” немного увеличиваем его (на 3 … 5 градусов), а на правом “холсте” — уменьшаем.

13. Создадим новую анимацию с размером кадра 1100 × 400. Нарисуем в первом кадре два прямоугольника с выключенным фоном. Размеры обоих прямоугольников: 550 × 400. Координаты первого прямоугольника приравняем нулю, координаты второго — значениям $X = 550$, $Y = 0$.

Преобразуем каждый прямоугольник в символ MovieClip (“холст”). Точку привязки обоих MovieClip разместим в верхнем левом углу. Дадим объектам имена экземпляров **xolst1** (левый) и **xolst2** (правый).




14. Присвоим первому кадру программный код (возьмем за основу программный код первого кадра из анимации построения 3D-графика в диметрической проекции, изменения показаны жирным шрифтом):

```
h0 = 400;
w0 = 550;
X0 = w0/2;
Y0 = h0/2;
AL = 45;
ALR = AL* Math.PI/180;
ALR1 = (AL+4)* Math.PI/180;
ALR2 = (AL-4)* Math.PI/180;
XMAX = 17;
YMAX = 17;
ZMAX = 1;
KZ = (Y0-10)/ZMAX;
KX = (w0-X0-10)/XMAX;
KY = Math.min(X0-10, h0-Y0-10)/Math.
cos(ALR)/YMAX;

X1 = -10;
X2 = 10;
DX = (X2-X1)/100;

Y1 = -10;
Y2 = 10;
DY = (Y2-Y1)/100;

function f(x,y) {
  r = Math.sqrt(x*x+y*y);
  if (r == 0) { r=0.001 }
  z = Math.sin(r)/r;
  return z;
}
```

```
Глобальные переменные:
– высота "холста";
– ширина "холста";
– координата X 3D системы координат
– координата Y 3D системы координат
– угол наклона оси y в градусах
– угол наклона оси y в радианах
– угол наклона оси y в радианах (левый)
– угол наклона оси y в радианах (правый)
– максимальное значение координаты x
– максимальное значение координаты y
– максимальное значение координаты z
– коэффициент масштаба по z
– коэффициент масштаба по x
– коэффициент масштаба по y


– диапазон изменения значений x
  (от X1 до X2)
– шаг изменения X

– диапазон изменения значений y
  (от Y1 до Y2)
– шаг изменения Y

Функция для построения графика:

– исключить деление на ноль
– вычисление значения функции
– возврат результата
```

Значения XMAX, YMAX и ZMAX подбираются вручную сразу для обоих кадров стереопары, чтобы графики полностью располагались внутри своих “холстов” и занимали максимум места на них.

15. Программный код, привязанный к левому объекту MovieClip (“холсту” **xolst1**), возьмем из анимации построения 3D-графика в диметрической проекции, изменения показаны жирным шрифтом:

```
onClipEvent(load) {
  this.lineStyle(1,0x888888,100);

  this.moveTo(_root.X0,_root.Y0);
  this.lineTo(_root.X0,_root.Y0-_root.KZ*_root.
ZMAX);

  this.moveTo(_root.X0,_root.Y0);
  this.lineTo(_root.X0+_root.KX*_root.XMAX,
              _root.Y0);

  this.moveTo(_root.X0,_root.Y0);
  this.lineTo(_root.X0-_root.KY*_root.YMAX*
Math.sin(_root.ALR1),_root.Y0+_root.KY*
_root.YMAX*Math.cos(_root.ALR1));

  this.lineStyle(1,0x0000FF,100);

  for (yy=_root.Y1; yy<=_root.Y2; yy+=_root.DY)
{
    for (xx=_root.X1; xx<=_root.X2; xx+=_root.DX) {

        zz = _root.f(xx,yy);
        x = _root.X0 + _root.KX*xx –
            _root.KY*yy*Math.sin(_root.ALR1);
        y = _root.Y0 + _
            root.KY*yy*Math.cos(_root.ALR1) –
            _root.KZ*zz;
        if (xx == _root.X1) { this.moveTo(x,y); }
            else { this.lineTo(x,y); }
    }
  }
}
```

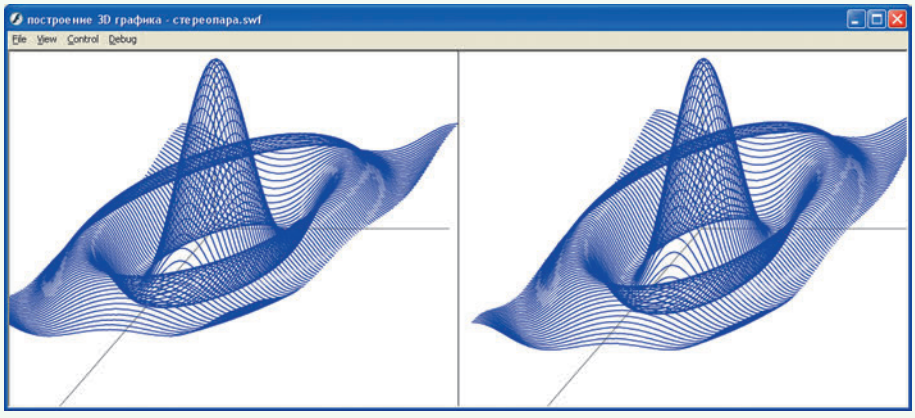
```
При запуске анимации:
– оси – серый цвет

– рисовать от точки (X0,Y0) ось Z

– рисовать от точки (X0,Y0) ось X

– рисовать от точки (X0,Y0)
  ось Y

– график синий толщиной 1 пикс.

– внешний цикл – перебор
  значений координаты y
– внешний цикл – перебор
  значений координаты x
– вычисление значения функции
– пересчет трехмерных координат
  в экранные

– первую точку – ставим,
  до остальных рисуем линию
```

16. Программный код, привязанный к правому объекту MovieClip (“холсту” **xolst2**), аналогичен, но вместо угла ALR1 используется угол ALR2:

```
onClipEvent(load) {

  this.lineStyle(1,0x888888,100);

  this.moveTo(_root.X0,_root.Y0);
  this.lineTo(_root.X0,_root.Y0-_root.KZ*_root.
ZMAX);

  this.moveTo(_root.X0,_root.Y0);
  this.lineTo(_root.X0+_root.KX*_root.XMAX,
              _root.Y0);

  this.moveTo(_root.X0,_root.Y0);
  this.lineTo(_root.X0-_root.KY*_root.YMAX*
        Math.sin(_root.ALR2),_root.Y0+_root.KY*
        _root.YMAX*Math.cos(_root.ALR2));

  this.lineStyle(1,0x0000FF,100);

  for (yy=_root.Y1; yy<=_root.Y2; yy+=_root.DY) {

  for (xx=_root.X1; xx<=_root.X2; xx+=_root.DX) {

     zz = _root.f(xx,yy);

       x = _root.X0 + _root.KX*xx -
           _root.KY*yy*Math.sin(_root.ALR2);
       y = _root.Y0 + _root.KY*yy*
           Math.cos(_root.ALR2) - _root.KZ*zz;

       if (xx==_root.X1) { this.moveTo(x,y); }
           else { this.lineTo(x,y); }
   }
  }

}
```

```
При запуске анимации:

- оси - серый цвет

- рисовать от точки (X0,Y0) ось Z

- рисовать от точки (X0,Y0) ось X

- рисовать от точки (X0,Y0)
  ось Y

- график синий толщиной 1 пикс.

- внешний цикл - перебор
  значений координаты y
- внешний цикл - перебор
  значений координаты x
- вычисление значения функции

- пересчет трехмерных координат
  в экранные

- первую точку - ставим,
  до остальных рисуем линию
```

17. Запустим анимацию. Проверим результат:




Для самостоятельного выполнения:

1. Постройте стереопары графиков функций, ранее предлагавшихся для построения в аксонометрии. (Выражение, определяющее функцию, и глобальные переменные, соответствующие диапазонам изменения координат *x* и *y*, а также максимальным значениям по осям, устанавливаются в программном коде первого кадра анимации однократно и действуют на оба кадра стереопары одновременно.)

2. Попытайтесь самостоятельно реализовать построение стереопары в аксонометрической проекции.

# УРОК 16 (САМОСТОЯТЕЛЬНАЯ ПРАКТИЧЕСКАЯ РАБОТА). СОЗДАНИЕ ИГРОВОЙ ПРОГРАММЫ

## 1. Используя все полученные ранее знания, создадим на Flash игру “Угадай число”.

**Сценарий игры:**

Компьютер “загадывает” (случайным образом) число от 1 до 100.

Пользователь (играющий) должен угадать “задуманное” компьютером число. Для этого пользователь вводит свое число (предполагая, что это — “задуманное” число). Компьютер в ответ выдает сообщение:

— “слишком много” — если введенное пользователем число больше “задуманного”;

— “слишком мало” — если введенное пользователем число меньше “задуманного”;

— “угадал” — если введенное пользователем число равно “задуманному”.

После того как число угадано, можно продолжить игру — компьютер “загадывает” следующее число, и т.д.

Каждый ввод пользователем числа считается за очередную попытку, количество попыток подсчитывается. При повторной игре определяется и выводится рекордное (наименьшее за все игры до закрытия Flash-ролика) количество попыток.

**Окно Flash-ролика и его основные элементы:**


***Клипы с именами mnogo, malo и ravno после создания и поименования нужно расположить точно друг поверх друга, чтобы эти надписи выводились в одном и том же месте кадра.***

**Сценарий работы программного кода:**

*Глобальные переменные:*

`NZ` — задуманное компьютером число (случайно от 1 до 100);

`N` — число пользователя, введенное в поле `userNumber`;

`k` — счетчик попыток;

`krec` — рекордное (минимальное) значение `k`.

*В начале работы Flash-ролика:*

Показать “кнопку” `starter`.

Скрыть элементы: `gotovo`, `userNumber`, `otvet`, `mnogo`, `malo`, `ravno`, `popyttxt`, `popyt`, `recordtxt`, `record`.

*При нажатии “кнопки” “Загадать число”:*

1) генерация случайного числа в `NZ`, `k` = 0;
2) скрыть надпись `ravno`;
3) показать элементы `gotovo`, `userNumber`, `otvet`, `popyttxt`, `popyt`;
4) вывод `k` в поле `popyt`;
5) если `krec` больше нуля, то:
   а) показать элементы `recordtxt`, `record`;
   б) вывод `krec` в `record`;
6) очистить поле `userNumber`;
7) спрятать “кнопку” `starter`.

*При нажатии кнопки “Есть”:*

1) чтение `N` из поля `userNumber`;
2) увеличение `k` на 1, вывод `k` в поле `popyt`;
3) сравнение `N` с `NZ`:
   а) если `N` < `NZ`, то вывод надписи `malo` и скрыть надписи `mnogo` и `ravno`;
   б) если `N` > `NZ`, то вывод надписи `mnogo` и скрыть надписи `malo` и `ravno`;

в) если N = NZ, то:

- вывод надписи ravno и скрыть надписи malo и mnogo;
- если krec = 0 ИЛИ k < krec, то krec = k, вывод krec в поле record, показать “кнопку” starter, скрыть “кнопку” otvet.

**Программный код**

*Код, привязанный к первому кадру анимации:*

```
k = 0; // счетчик попыток
krec = 0; // рекорд (минимальное k)
N = 0; // число игрока
NZ = 0;  // загаданное компьютером число от 1 до 100

starter._visible = true;
gotovo._visible = false;
userNumber._visible = false;
otvet._visible = false;
mnogo._visible = false;
malo._visible = false;
ravno._visible = false;
popyttxt._visible = false;
popyt._visible = false;
recordtxt._visible = false;
record._visible = false;
```

```
Глобальные переменные

Показ и скрытие элементов на
кадре в начале работы анимации
```

*Код, привязанный к “кнопке” “Загадать число” (starter):*

```
on (press) {
  _root.NZ = Math.round(99*Math.random()-1);
  _root.userNumber.text = '';
  _root.k = 0;
  _root.ravno._visible = false;
  _root.gotovo._visible = true;
  _root.userNumber._visible = true;
  _root.otvet._visible = true;
  _root.popyttxt._visible = true;
  _root.popyt._visible = true;
  _root.popyt.text = _root.k;
  if (_root.krec > 0)  {
       _root.recordtxt._visible = true;
       _root.record._visible = true;
       _root.record.text = _root.krec;
  }
  _root.starter._visible = false;
}
```

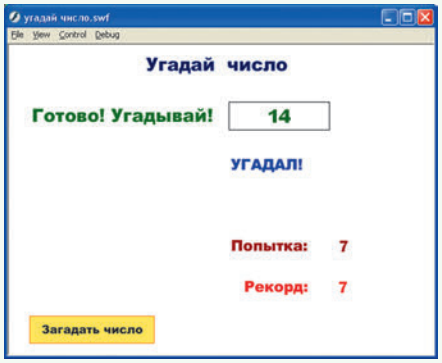
```
При нажатии:
– задумать число
– очистить поле ввода userNumber
– обнулить счетчик попыток
– показ и скрытие элементов на кадре
для старта очередной игры

– вывод номера попытки
– вывод рекорда (кроме самой первой
игры, когда рекорд еще не определен)

– скрыть "кнопку" "Загадать число"
```

*Код, привязанный к “кнопке” “Есть!” (otvet):*

```
on (press) {
  _root.N = _root.userNumber.text;
  _root.k += 1;
  _root.popyt.text = _root.k;
  if (_root.N < _root.NZ) {
       _root.malo._visible = true;
       _root.mnogo._visible = false;
       _root.ravno._visible = false;
  }
  if (_root.N > _root.NZ) {
       _root.malo._visible = false;
       _root.mnogo._visible = true;
       _root.ravno._visible = false;
  }
  if (_root.N == _root.NZ) {
       _root.otvet._visible = false;
       _root.malo._visible = false;
       _root.mnogo._visible = false;
       _root.ravno._visible = true;
       if ((_root.krec == 0) ||
           (_root.k < _root.krec)) {
            _root.krec = _root.k;
       }
       _root.record.text = _root.krec;
       _root.starter._visible = true;
  }
}
```

```
При нажатии:
– считать число пользователя
– увеличить номер попытки
– вывести номер попытки
– если введенное число меньше задуманного,
то сообщить об этом

– если введенное число больше задуманного,
то сообщить об этом

– если введенное число равно загаданному, то:
  – скрыть кнопку "Есть"
  – сообщить, что число угадано

  – определить рекорд: в первой игре это
просто k, а далее – минимальное из всех k

  – вывод рекорда попыток
  – вывести кнопку "Загадать число"
    для начала следующей игры
```

**Вид кадра в процессе игры:**




## 2. Создадим на Flash игру “Быки и коровы”.

**Сценарий игры:**

Компьютер “загадывает” (случайным образом) четырехзначное число, все цифры которого различны и могут быть от 0 до 9 (в том числе ноль может быть первой цифрой).

Пользователь (играющий) должен угадать “задуманное” компьютером число. Для этого пользователь вводит свое число (предполагая, что это — “задуманное” число). Компьютер в ответ выдает информацию в виде “ … быков, … коров”, где:

— количество “быков” — это количество цифр, которые во введенном пользователем числе указаны на правильных местах;

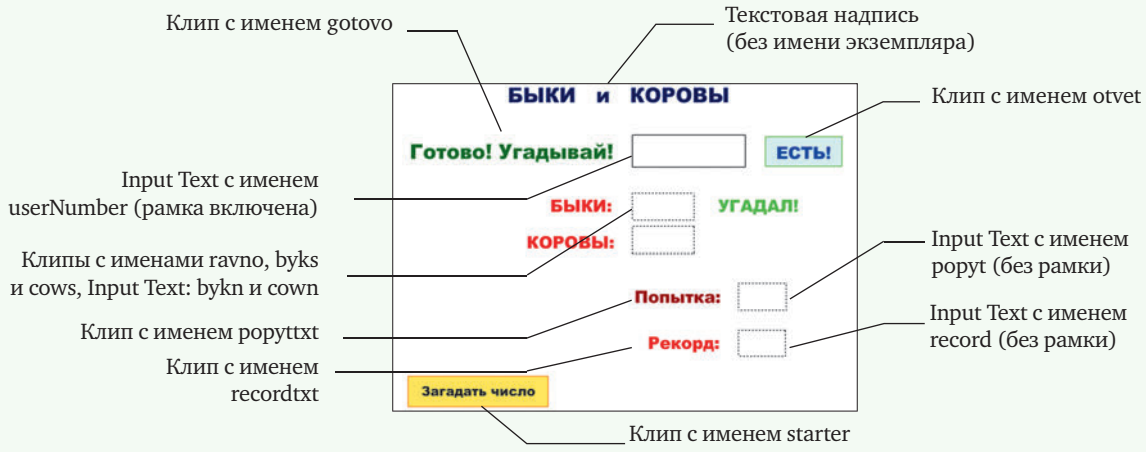
— количество “коров” — это количество цифр, которые есть в задуманном числе, но во введенном пользователем числе указаны не на своих местах.

(Например, если задумано число 1234, а пользователь ввел число 2534, то компьютер ответит “2 быка и 1 корова”, где “2 быка” — это цифры 3 и 4 в конце обоих чисел, а “1 корова” — это цифра 2.)

После того как число угадано, можно продолжить игру — компьютер “загадывает” следующее число, и т.д.

Каждый ввод пользователем числа считается за очередную попытку, количество попыток подсчитывается. При повторной игре определяется и выводится рекордное (наименьшее за все игры до закрытия Flash-ролика) количество попыток.

**Окно Flash-ролика и его основные элементы:**






**Сценарий работы программного кода:**

*Глобальные переменные:*

`a1`, `a2`, `a3`, `a4` — задуманные компьютером четыре цифры числа (с контролем отсутствия повторов), само задуманное число представляется как последовательность этих цифр (`a1a2a3a4`);

`N` — число пользователя, введенное в поле `userNumber`, раскладывается на отдельные цифры `n1`, `n2`, `n3`, `n4` для попарного сравнения с `a1`, `a2`, `a3`, `a4`;

`kb` — счетчик количества “быков”;

`kc` — счетчик количества “коров”;

`k` — счетчик попыток;

`krec` — рекордное (минимальное) значение `k`.

*В начале работы Flash-ролика:*

Показать “кнопку” `starter`.

Скрыть элементы: gotovo, userNumber, otvet, ravno, byks, cows, bykn, cown, popyttxt, popyt, recordtxt, record.

*При нажатии “кнопки” “Загадать число”:*

1) генерация четырех цифр числа в a1, a2, a3, a4 (с контролем отсутствия повторов), k = 0;
2) скрыть надпись ravno;
3) показать элементы gotovo, userNumber, otvet, byks, cows, bykn, cown, popyttxt, popyt;
4) обнуление k и вывод k в поле popyt;
5) обнуление kb и вывод kb в поле bykn;
6) обнуление kc и вывод kc в поле cown;
7) если krec больше нуля, то:
   а) показать элементы recordtxt, record;
   б) вывод krec в record;
8) очистить поле userNumber;
9) спрятать “кнопку” starter.

*При нажатии кнопки “Есть”:*

1) чтение N из поля userNumber;
2) увеличение k на 1, вывод k в поле popyt;
3) разбивка N на цифры n1, n2, n3, n4;
4) сравнение a1 с n1, a2 с n2, a3 с n3, a4 с n4 и подсчет количества равенств в счетчике kb, вывод kb в поле bykn;
5) сравнение a1 с n2, n3, n4, a2 с n1, n3, n4, a3 с n1, n2, n4, a4 с n1, n2, n3 и подсчет количества равенств в счетчике kc, вывод kc в поле cown;
6) если kb = 4, то
   • вывод надписи ravno;
   • если krec = 0 ИЛИ k < krec, то krec = k, вывод krec в поле record, показать “кнопку” starter, скрыть “кнопку” otvet.

**Программный код**

*Код, привязанный к первому кадру анимации:*

```
k = 0; // счетчик попыток
krec = 0; // рекорд (минимальное k)
kb = 0;  // счетчик "быков"
kc = 0;  // счетчик "коров"
N = ''; // число игрока (считывается как текстовая строка)
a1 = 0;  // цифры загаданного числа
a2 = 0;
a3 = 0;
a4 = 0;
n1 = 0;  // цифры введенного числа
n2 = 0;
n3 = 0;
n4 = 0;

starter._visible = true;
gotovo._visible = false;
userNumber._visible = false;
otvet._visible = false;
byks._visible = false;
bykn._visible = false;
cows._visible = false;
cown._visible = false;
ravno._visible = false;
popyttxt._visible = false;
popyt._visible = false;
recordtxt._visible = false;
record._visible = false;
```

Глобальные переменные

Показ и скрытие элементов на кадре в начале работы анимации

*Код, привязанный к “кнопке” “Загадать число” (starter):*

```
on (press) {
  _root.a1 = Math.round(9*Math.random());
  do { _root.a2 = Math.round(9*Math.random());
   } while (_root.a2 == _root.a1);
  do { _root.a3 = Math.round(9*Math.random());
   } while ((_root.a3 == _root.a1) ||
            (_root.a3 == _root.a2));
  do { _root.a4 = Math.round(9*Math.random());
   } while ((_root.a4 == _root.a1) ||
            (_root.a4 == _root.a2) ||
            (_root.a4 == _root.a3));

  _root.userNumber.text = '';
  _root.k = 0;
  _root.kb = 0;
  _root.kc = 0;
  _root.ravno._visible = false;
  _root.gotovo._visible = true;
  _root.userNumber._visible = true;
  _root.otvet._visible = true;
  _root.byks._visible = true;
  _root.bykn.text = _root.kb;
  _root.bykn._visible = true;
  _root.cows._visible = true;
  _root.cown.text = _root.kc;
  _root.cown._visible = true;
  _root.popyttxt._visible = true;
  _root.popyt._visible = true;
  _root.popyt.text = _root.k;
  if (_root.krec > 0)  {
    _root.recordtxt._visible = true;
    _root.record._visible = true;
    _root.record.text = _root.krec;
  }
  _root.starter._visible = false;
}
```

```
При нажатии:
— 1-я случайная цифра

2-я случайная цифра —
  искать, пока она
  не будет не равна 1-й
3-я случайная цифра —
  искать, пока она
  не будет не равна 1-й и 2-й

4-я случайная цифра — искать, пока
она не будет не равна 1-й, 2-й и 3-й
— очистить поле ввода userNumber
— обнулить счетчик попыток
— обнулить счетчик "быков"
— обнулить счетчик "коров"
— показ и скрытие элементов на кадре
для старта очередной игры

— вывод номера попытки
— вывод рекорда (кроме самой первой
игры, когда рекорд еще не определен)

— скрыть "кнопку" "Загадать число"
```

*Код, привязанный к “кнопке” “Есть!” (otvet):*

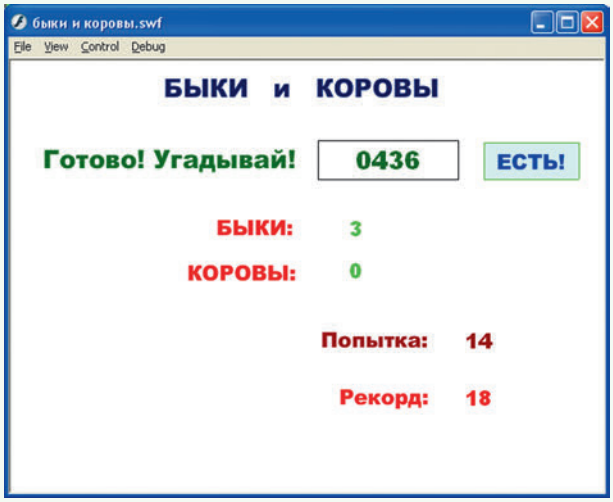
```
on (press) {
  _root.N = _root.userNumber.text;
  _root.k += 1;
  _root.popyt.text = _root.k;

  _root.n1 = parseInt(_root.N.substr(0,1));
  _root.n2 = parseInt(_root.N.substr(1,1));
  _root.n3 = parseInt(_root.N.substr(2,1));
  _root.n4 = parseInt(_root.N.substr(3,1));

  _root.kb = 0;
  if (_root.a1 == _root.n1) { _root.kb += 1; }
  if (_root.a2 == _root.n2) { _root.kb += 1; }
  if (_root.a3 == _root.n3) { _root.kb += 1; }
  if (_root.a4 == _root.n4) { _root.kb += 1; }
  _root.bykn.text = _root.kb;
  _root.kc = 0;
  if ((_root.a1 == _root.n2) ||
      (_root.a1 == _root.n3) ||
      (_root.a1 == _root.n4)) { _root.kc += 1; }
  if ((_root.a2 == _root.n1) ||
      (_root.a2 == _root.n3) ||
      (_root.a2 == _root.n4)) { _root.kc += 1; }
  if ((_root.a3 == _root.n1) ||
      (_root.a3 == _root.n2) ||
      (_root.a3 == _root.n4)) { _root.kc += 1; }
  if ((_root.a4 == _root.n1) ||
      (_root.a4 == _root.n2) ||
      (_root.a4 == _root.n3)) { _root.kc += 1; }
  _root.cown.text = _root.kc;
```

```
При нажатии:
— считать число пользователя
— увеличить номер попытки
— вывести номер попытки

— выделить из числа 1-ю цифру
— выделить из числа 2-ю цифру
— выделить из числа 3-ю цифру
— выделить из числа 4-ю цифру

— считать количество "быков":
  каждое совпадение цифр в одной
  и той же позиции — это один "бык"

— вывести количество "быков"
— считать количество "коров":
  совпадение цифр хотя бы в одной
  "не своей" позиции — это
  одна "корова"

— вывести количество "быков"
```

```
  if (_root.kb == 4) {

    _root.otvet._visible = false;
    _root.ravno._visible = true;
    if ((_root.krec == 0) || (_root.k < _root.
krec)) {
      _root.krec = _root.k;
    }
    _root.record.text = _root.krec;
    _root.starter._visible = true;
  }
}
```

```
— если все 4 цифры введенного числа
совпали с загаданными, то:
  — скрыть кнопку "Есть"
  — сообщить, что число угадано
  — определить рекорд: в первой игре
это просто k, а далее — минимальное
из всех k

  — вывод рекорда попыток
  — вывести кнопку "Загадать число"
для начала следующей игры
```

**Вид кадра в процессе игры:**

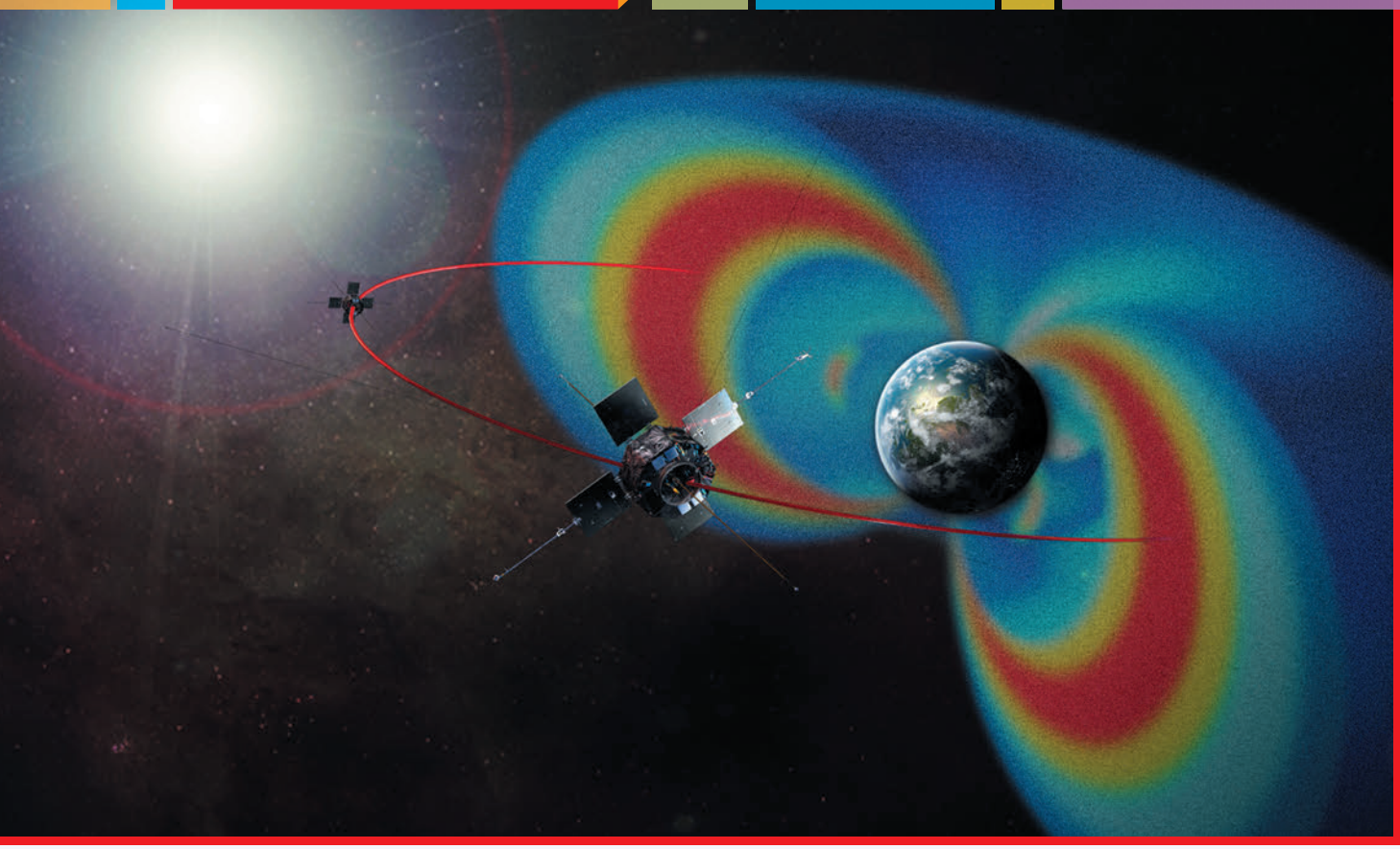



*Задание для самостоятельного выполнения:*

Реализуйте игру “Быки и коровы” с использованием массивов для записи цифр задуманного и введенного чисел.

Секрет игры

Чтобы угадать число, сначала надо пытаться получить 4 “коровы”, а потом перестановками цифр числа получить из них 4 “быка”.



# Методика выполнения задания 27 из демонстрационного варианта ЕГЭ по информатике 2016 года

**Д.М. Златопольский,**
Москва

*Условие*

"В физической лаборатории проводится долговременный эксперимент по изучению радиационного поля Земли. По каналу связи каждую минуту в лабораторию передается положительное целое число — текущее показание прибора "Сигма-2015". Количество передаваемых чисел известно и не превышает 10 000. Все числа не превышают 1000. Временем, в течение которого происходит передача, можно пренебречь.

Необходимо вычислить "бета-значение" серии показаний прибора — минимальное **четное** произведение двух показаний, между моментами передачи которых прошло не менее шести минут. Если получить такое произведение не удается, ответ считается равным –1.

*Вам предлагаются два задания, связанных с этой задачей: задание А и задание Б. Вы можете решать оба задания или одно из них по своему выбору.*

*Итоговая оценка выставляется как* ***максимальная*** *из оценок за задания А и Б. Если решение одного из заданий не представлено, то считается, что оценка за это задание — 0 баллов.*

*Задание Б является усложненным вариантом задания А, оно содержит дополнительные требования к программе.*

Фото с сайта *www.nasa.gov*

А. Напишите на любом языке программирования программу для решения поставленной задачи, в которой входные данные будут запоминаться в массиве, после чего будут проверены все возможные пары элементов. Перед программой укажите версию языка программирования.

ОБЯЗАТЕЛЬНО укажите, что программа является решением ЗАДАНИЯ А. Максимальная оценка за выполнение задания А — 2 балла.

Б. Напишите программу для решения поставленной задачи, которая будет эффективна как по времени, так и по памяти (или хотя бы по одной из этих характеристик).

Программа считается эффективной по времени, если время работы программы пропорционально количеству полученных показаний прибора $N$, т.е. при увеличении $N$ в $k$ раз время работы программы должно увеличиваться не более чем в $k$ раз.

Программа считается эффективной по памяти, если размер памяти, использованной в программе для хранения данных, не зависит от числа $N$ и не превышает 1 килобайта.

Перед программой укажите версию языка программирования и кратко опишите использованный алгоритм.

ОБЯЗАТЕЛЬНО укажите, что программа является решением ЗАДАНИЯ Б. Максимальная оценка за правильную программу, эффективную по времени и по памяти, — 4 балла.

Максимальная оценка за правильную программу, эффективную по времени, но неэффективную по памяти, — 3 балла.

НАПОМИНАЕМ! Не забудьте указать, к какому заданию относится каждая из представленных вами программ.

Входные данные представлены следующим образом. В первой строке задается число $N$ — общее количество показаний прибора. Гарантируется, что $N > 6$. В каждой из следующих $N$ строк задается одно положительное целое число — очередное показание прибора.

*Пример входных данных:*

```
11
12
45
5.5
3
17

23
21
20
19
18
17
```

Программа должна вывести одно число — описанное в условии произведение либо –1, если получить такое произведение не удается.

*Пример выходных данных для приведенного выше примера входных данных:*

```
54
```

”.

Начнем с решения задания А, как более простого[1].

Здесь метод решения задачи указан в условии применительно к заданию А — сохранить все показания прибора в массиве, после чего:

1) рассмотреть все возможные пары значений;

2) проверить произведение чисел каждой пары на четность; в случае четности сравнить его с минимальным четным произведением среди уже рассмотренных пар. Если очередное произведение меньше минимального, то в качестве последнего должно быть принято это очередное произведение.

Пары должны образовывать только показания, между моментами передачи которых прошло не менее шести минут. Так, для примера, показанного на *рис.* 1, для показания 12,4 нужно рассматривать не все предыдущие значения, а только оттененные.

В программе, реализующей описанный метод решения, кроме величины $N$ используем следующие основные величины:

— *м_пр* — искомое значение;

— *п* — массив для хранения показаний прибора.

Программа на школьном алгоритмическом языке:

```
алг Задание_27А
нач цел N, пр, м_пр, i, j
  |Ввод количества показаний прибора
  |значений N
ввод N
  |Описание массива значений п
цел таб п[1:N]
  |Ввод всех показаний и сохранение
  |их в массиве п
  нц для i от 1 до N
  ввод п[i]
кон
  |Поиск ответа
  |Начальное значение минимального
  |произведения
м_пр := 1000 * 1000 + 1
  |Перебор всех пар значений
  |Для каждого значения, начиная с 7-го
нц для i от 7 до N
    |и для каждого допустимого
    |предшествующего значения
    нц для j от 1 до i - 6
```

| **Номер показания** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Очередное показание** | 14,4 | 3,2 | 23,0 | 17,2 | 5,8 | 2,2 | 67,0 | 44,3 | 11,2 | **12,4** | … |
| | | … | $i-7$ | $i-6$ | $i-5$ | $i-4$ | $i-3$ | $i-2$ | $i-1$ | $i$ | |

*Рис.* 1

[1] В [1] разбор решения начинается с самого сложного варианта решения задания Б.

```
        |рассчитываем произведение пары
        |чисел
        пр := п[i] * п[j]
        |проверяем их произведение
        |на четность
        если mod(пр, 2) = 0
          то
           |сравниваем их произведение
           |со "старым" значением
           |величины м_пр
           если пр < м_пр
    то    |Встретилась пара чисел
          |с минимальным произведением
                |Меняем значение м_пр
    м_пр := пр
    все
все
кц
кц
|Вывод ответа
если м_пр := 1000 * 1000 + 1
  |Если минимальное произведение
  |не изменилось
  то
    |минимальное четное произведение
    |показаний
    |получить не удалось
    м_пр := -1
все
вывод нс, м_пр
кон
```

Можно также сравнение произведения со “старым” минимальным значением совместить с проверкой на четность, используя сложное (составное) условие[2]:

```
нц для i от 7 до N
  |и для каждого допустимого
  |предшествующего значения
  нц для j от 1 до i - 6
     |рассчитываем произведение пары
     пр := п[i] * п[j]
     |проверяем произведение
     если mod(пр, 2) = 0 и пр < м_пр
       то
        |меняем значение м_пр
        м_пр := пр
     все
  кц
кц
|Вывод ответа
...
```

Обратим внимание, что описание массива в теле программы, после задания значения *N*, как это сделано в приведенном случае, допустимо не во всех языках программирования. Кроме того, в школьном алгоритмическом языке (система КуМир) для проверки и возможного изменения величины *м_пр* (и в большинстве случаев в других программах, приведенных в статье далее) можно использовать стандартную функцию *min*, возвращающую минимальное значение двух ее аргументов:

```
...
|сравниваем их произведение со "старым"
|значением величины м_пр
м_пр := min(п[i] * п[j], м_пр)
...
```

Заметим также, что в приведенной в [1] программе решения задания А на языке Паскаль (она обозначена как *Программа 4*):

1) вместо значения 6 (требуемое расстояние между двумя показаниями) используется константа *s*;

2) величина *пр* не используется (дважды записывается произведение пары показаний);

3) перебор всех возможных пар значений проводится следующим образом:

```
for i := 1 to N - s do
  begin
    for j := i + s to N - s do
      begin
        if (a[i] * a[j] mod 2 = 0) and
           (a[i] * a[j] < m)
        then m := a[i] * a[j]
      end;
    end;
```

или на школьном алгоритмическом языке и при принятых именах величин:

```
нц для i от 1 до N - 6
  нц для j от i + 6 до N
    если mod(п[i] * п[j], 2) = 0 и
         п[i] * п[j] < м_пр
      то
        м_пр := п[i] * п[j]
    все
  кц
кц
```

что по сути аналогично варианту, приведенному выше[3].

**Примечание**. Использование в программе на Паскале двух составных операторов является излишним. Можно применить такой, более компактный, вариант фрагмента:

```
for i := 1 to N - s do
  for j := i + s to N - s do
    if …
```

Хотя задание А мы решили, обсудим одно из направлений совершенствования разработанной программы, которое будет полезно при решении задачи В. Оно заключается в следующем.

Разработанная выше программа является неэффективной по времени ее выполнения из-за большого числа сравнений (оно пропорционально $N^2$). Этот

[2] В этом случае в системах программирования, в которых проводится полная проверка сложных условий, сравнение произведения со “старым” значением будет происходить и для нечетных произведений.

[3] В программах, в которых массив со всеми показаниями прибора не используется (см. далее), приведенный в [1] способ формирования пар значений невозможен.

недостаток можно ликвидировать, если для каждого вводимого показания знать минимальное показание среди предыдущих, отстоящих от него не менее чем на 6 значений. Эти минимальные показания можно хранить в специальном массиве, заполнять который можно по мере ввода показаний прибора[4].

Основные этапы решения задачи по такому методу:

1) ввод и запоминание в массиве *п* первых шести показаний и заполнение первых шести элементов массива *мин_п*;

2) присваивание начального значения величины *м_пр*;

3) ввод остальных показаний (элементов массива *п*) и для каждого из них:

— уточнение при необходимости очередного значения массива *мин_п*;

— проверка произведения очередного показания на минимальное значение "предыдущих" (отстоящих от него на шесть значений). Если оно четное и меньше "старого" значения искомого произведения (*мин_п*), то изменение значения *мин_п*;

4) вывод ответа (значения *мин_п* или –1).

Соответствующая программа:

```
алг Задание_27_2
нач цел N, вещ м_пр, цел i
  ввод N
  |Описание массивов значений п
  вещ таб п[1:N]
  |и мин_п (минимальные значения среди
  |заданных)
  вещ таб мин_п[1:N]
  |Ввод первых шести показаний
  |1-е показание (и одновременно
  |значение 1-го элемента массива мин_п)
  ввод мин_п[1]
  |2..6 показания
  нц для i от 2 до 6
    ввод п[i]
    |Заполняем массив мин_п
    если п[i] < мин_п[i - 1]
    |Можно также использовать функцию min
      то
        мин_п[i] := п[i]
      иначе
        мин_п[i] := мин_п[i - 1]
    все
  кц
  |Поиск ответа
  м_пр := 1000 * 1000 + 1
  |Ввод остальных показаний
  нц для i от 7 до N
    ввод п[i]
    |Заполняем массив мин_п
   |(можно также использовать функцию min)
    если п[i] < мин_п[i - 1]
      то
        мин_п[i] := п[i]
      иначе
        мин_п[i] := мин_п[i - 1]
    все
    |Проверяем произведение очередного
    |показания
    |на минимальное значение "предыдущих"
    если mod(п[i] * мин_п[i - 6], 2) = 0
         и п[i] * мин_п[i - 6] < м_пр
      то |Встретилась пара чисел с
         |минимальным четным произведением
        м_пр := п[i] * мин_п[i - 6]
    все
  кц
  |Вывод ответа
  … (см. выше)
кон
```

Приведенный вариант программы более эффективен по времени, чем программа Задание_27А, но менее эффективен по памяти, так как использует два массива. Последний недостаток можно устранить, если массив *мин_п* не использовать.

Обсудим вопрос — какое значение следует использовать "в паре" с очередным введенным значением *п*[*i*] и сравнивать их произведение со "старым" значением искомой величины *м_пр*? Ответ на этот вопрос дает анализ *рис.* 1. Для очередного значения, равного 12,4, следует использовать минимальное из оттененных значений — 3,2 (их произведение равно 39,68). Значит, нужно иметь информацию о минимальном значении среди уже заданных показаний, причем не всех, а отстоящих от очередного введенного показания не менее чем на шесть значений. Величине, хранящей информацию о таком минимальном значении, в программе дадим имя *мин_п*.

Основные этапы решения задачи по описанному методу:

1) ввод и запоминание в массиве *п* первых шести показаний;

2) присваивание начальных значений величин *мин_п* и *м_пр*;

3) ввод остальных показаний (элементов массива *п*) и для каждого из них:

— уточнение при необходимости значения *мин_п* (путем сравнения "старого" значения *мин_п* и элемента массива п, отстоящего от очередного введенного на шесть показаний);

— проверка произведения очередного показания на минимальное значение "предыдущих". Если оно четное и меньше "старого" значения искомого произведения (*мин_п*), то изменение значения *мин_п*;

4) вывод ответа (значения *мин_п* или –1).

Соответствующая программа:

```
алг Задание_27_3
нач цел N, вещ мин_п, м_пр, цел i
  ввод N
```

[4] Значения элементов этого массива будут образовывать невозрастающую последовательность (убедитесь в этом!).

```
  |Описание массива значений п
  вещ таб п[1:N]
  |Ввод первых шести показаний
  нц для i от 1 до 6
    ввод п[i]
  кц
 |Поиск ответа
  мин_п := 1000
  м_пр := 1000 * 1000 + 1
  |Ввод остальных показаний
  нц для i от 7 до N
    ввод п[i]
    |Проверяем значение мин_п
    |(обратите внимание на индекс
      элемента массива п)
    если п[i - 6] < мин_п
      то
        мин_п := п[i - 6]
    все
    |Проверяем произведение очередного
    |показания на минимальное
    |значение "предыдущих"
    если mod(п[i] * мин_п, 2) = 0 и
         п[i] * мин_п < м_пр
      то
        м_пр := п[i] * мин_п
    все
  кц
  |Вывод ответа
  …
кон
```

Можно также не использовать массив показаний *п*, а сразу обрабатывать очередное введенное значение (“сохранив” при этом массив *мин_п*). Прежде чем представлять программу, заметим, что очередное показание имеет в ней имя *п*.

```
алг Задание_27_4
нач цел N, вещ п, м_пр, цел i
  ввод N
  |Описание массива мин_п
  |(минимальные значения среди заданных)
вещ таб мин_п[1:N][5]
  |Ввод первых шести показаний
  |1-е показание (и одновременно
  |значение 1-го элемента массива мин_п)
  ввод мин_п[1]
  |2..6 показания
  нц для i от 2 до 6
    ввод п
    |Заполняем массив мин_п
    если п < мин_п[i - 1]
      то
        мин_п[i] := п
      иначе
        мин_п[i] := мин_п[i - 1]
    все
  кц
```

[5] Использование массива *мин_п* размером *N* – 6 невозможно в случае, когда 12 > *N* > 6.

```
|Поиск ответа
  м_пр := 1000 * 1000 + 1
  |Ввод остальных показаний
  нц для i от 7 до N
    ввод п
    |Заполняем массив мин_п
    если п < мин_п[i - 1]
      то
        мин_п[i] := п
      иначе
        мин_п[i] := мин_п[i - 1]
    все
    |Сравниваем произведение
   |(можно также использовать функцию min)
    если mod(п * мин_п[i - 6], 2) = 0 и
         п * мин_п[i - 6] < м_пр
      то
        м_пр := п * мин_п[i - 6]
    все
  кц
  |Вывод ответа
  …
кон
```

После разработки программ `Задание_27_3` и `Задание_27_4` мы подошли к решению задания B.

Прежде всего здесь при выполнении задания следует обратить внимание на комментарий в условии, касающийся эффективности программы по памяти: “Программа считается эффективной по памяти, если размер памяти, использованной в программе для хранения данных, не зависит от числа *N* и не превышает 1 килобайта”.

Этот комментарий говорит о том, что использование массива для хранения *всех* показаний прибора в данном случае невозможно. Вариант программы без этого массива мы только что разработали. Осталось “отказаться” от массива *мин_п*. Как?

Выше уже отмечалось, что значения элементов этого массива образуют невозрастающую последовательность. Например, для следующих показаний прибора:

45, 53, 33, 46, 47, 15, 26, 15, 44, 14, 13, 37, 22, …

— массив *мин_п* будет иметь вид:

| 45 | 45 | 33 | 33 | 33 | 15 | 15 | 15 | 15 | 14 | 13 | 13 | 13 | … |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* | *11* | *12* | *13* | … |

Первый элемент этого массива будет “использован” седьмым показанием прибора (равным 26). И больше не понадобится! И вообще, какие из элементов этого массива надо знать в момент задания некоторого очередного показания прибора, например, 11-го (равного 13)? Ответ показан на *рис.* 2.

| 45 | 45 | 33 | 33 | 33 | 15 | 15 | 15 | 15 | 14 | **13** | 13 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *1* | *2* | *3* | *4* | *5* | *6* | *7* | *8* | *9* | *10* | *11* | *12* | *13* |

*Рис.* 2

А после ввода 11-го показания “ненужным” будет 5-й элемент (33), но станет “нужным” (для последующих расчетов) введенное значение 13. Следовательно, вместо массива размером $N - 6$ элементов достаточно использовать только массив из шести элементов, меняя его содержание по мере ввода новых показаний[6]. Для приведенного примера особенности изменения этого массива показаны на *рис.* 3.

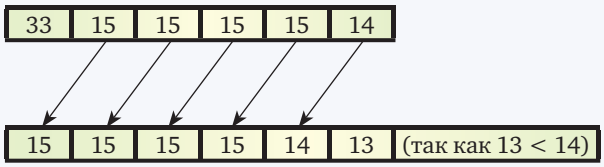



*Рис.* 3

Итак, идея решения — использовать массив из шести элементов, хранящий минимальные значения показаний среди отстоящих от текущего показания не менее чем на шесть значений, и менять его содержание по мере ввода новых показаний.

*Этапы соответствующего алгоритма*

1. Ввод значения *N*.
2. Ввод первых шести показаний и сохранение их в массиве.
3. Для каждого показания, начиная с 7-го:
3.1. Ввод очередного показания.
3.2. Расчет произведения этого показания на 1-й элемент указанного массива и сравнение произведения с минимальным произведением уже рассмотренных пар.
3.3. Изменение массива минимальных значений.
4. Вывод ответа.

Соответствующая программа:

```
алг Задание_27Б
нач цел N, вещ п, м_пр, вещ таб мин_п[1:6],
    цел i, j
  ввод N
  |Ввод первых шести показаний
  |1-е показание (и одновременно
  |значение 1-го элемента массива мин_п)
  ввод мин_п[1]
  |2..6 показания
  нц для i от 2 до 6
    ввод п
    |Заполняем массив мин_п
    если п < мин_п[i - 1]
      то
        мин_п[i] := п
      иначе
        мин_п[i] := мин_п[i - 1]
    все
  кц
  |Поиск ответа
  м_пр := 1000 * 1000 + 1
  |Ввод остальных показаний
  нц для i от 7 до N
    ввод п
    |Сравниваем произведение
    |(обратите внимание на индекс элемента
    |массива)
    если п * мин_п[1] < м_пр
      то
        м_пр := п * мин_п[1]
    все
    |Меняем массив мин_п
    |Сдвигаем влево 2..6 элементы
    нц для j от 1 до 5
      мин_п[j] := мин_п[j + 1]
    кц
    |При необходимости (!) меняем и
    |6-й элемент
    если п < мин_п[6]
      то
        мин_п[6] := п
    все
  кц
  |Вывод ответа
  ...
кон
```

Можно также использовать массив не минимальных значений, а меняющийся массив из шести последних показаний. В приведенном ниже варианте программы имя этого массива — *п*2.

При этом понадобится также величина, хранящая минимальное введенное число с учетом требуемого “расстояния” между показаниями (имя величины — *мин_п*). Ее значение сравнивается с первым элементом массива *п*2.

Особенности изменения массива *п*2 и значения величины *мин_п* покажем на примере следующих показаний прибора:

45, 53, 33, 46, 47, 15, 26, 18, 44, 14, 13, 37, 22, …

Перед вводом показания 37 массив *п*2 будет иметь вид:

| 15 | 26 | 18 | 44 | 14 | 13 |
|---|---|---|---|---|---|

При этом *мин_п* = 33.

После ввода показания, равного 37, для обработки станет доступным первый элемент массива (15). Его сравнение со “старым” значением величины *мин_п* показывает, что новым минимумом станет число 15, которое и должно использоваться при расчете произведения. Новый вариант массива *п*2 для дальнейшей обработки показан на *рис.* 4.

| 26 | 18 | 44 | 14 | 13 | 37 |
|---|---|---|---|---|---|

*Рис.* 4

*Этапы соответствующего алгоритма*

1. Ввод значения *N*.
2. Ввод первых шести показаний и сохранение их в массиве.
3. Для каждого показания, начиная с 7-го:
3.1. Ввод очередного показания.

[6] Можно вместо массива из шести элементов применить шесть “простых” величин, однако использование массива упрощает изменение значений по мере ввода новых показаний (см. программу далее).

3.2. Уточнение (при необходимости) минимального показания *мин_п*.

3.3. Расчет произведения очередного показания на *мин_п* и сравнение произведения с минимальным произведением уже рассмотренных пар.

3.4. Изменение массива значений.

4. Вывод ответа.

Программа, работающая по последнему алгоритму, имеет вид:

```
алг Задание_27Б_вариант_2
нач цел N, вещ п, мин_п, м_пр,
    вещ таб п2[1:6], цел i, j
  ввод N
  м_пр := 1000 * 1000 + 1
мин_п := 1000
  |Ввод первых шести показаний
  |(и одновременно заполнение массива п2)
  нц для i от 1 до 6
    ввод п2[i]
  кц
  |Ввод остальных показаний
  нц для i от 7 до N
    ввод п
    |Уточняем при необходимости
    |значение мин_п (обратите внимание
    |на индекс элемента массива)
    если п2[1] < мин_п
      то
        мин_п := п2[1]
    все
    |Сравниваем произведение
    если mod(п * мин_п, 2) = 0 и
         п * мин_п < м_пр
      то
        м_пр := п * мин_п
    все
    |Меняем массив п2 (см. рис. 4)
    |Сдвигаем влево 2..6 элементы
    нц для j от 1 до 5
      п2[j] := п2[j + 1]
    кц
    |Записываем в массив очередное
    |показание
    п2[6] := п
  кц
  |Вывод ответа
  …
кон
```

В заключение заметим, что в [1] представлена программа решения задачи Б (обозначенная как *Программа 1*), аналогичная программе Задание_27Б, но отличающаяся тем, что изменение массива *мин_п* происходит не путем постоянного сдвига его элементов, а (цитата) “путем циклического заполнения массива”. Приводя соответствующую программу, оценить ее понятность предлагаем читателям.

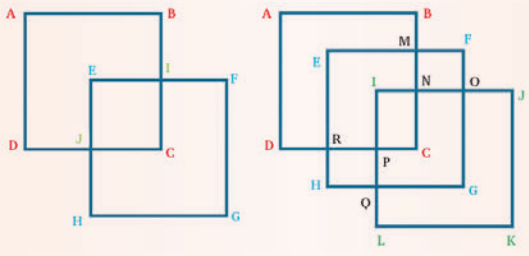
```
алг
нач
  цел s = 6  |требуемое расстояние
             |между показаниями
  цел N
  ввод N
  вещ a  | очередное показание прибора
  вещтаб мини [0:s-1] |текущие минимумы
                      | последних s элементов
  цел i
  |вводим первые s показаний,
  |фиксируем минимумы
  ввод мини[1]
  нц для i от 2 до s
    ввод a
    мини[mod(i, s)] := min(a, мини[i-1]
  кц
  вещ м | минимальное значение произведения
  м = 1000 * 1000 + 1
  нц для i от s+1 до N
    ввод a
   мини[mod(i, s)] := min(a, мини[mod(i-1, s)]
  кц
  вывод м
кон
```

## Литература

1. Демонстрационный вариант контрольно-измерительных материалов единого государственного экзамена 2016 года по информатике и ИКТ. Подготовлен Федеральным государственным бюджетным научным учреждением “Федеральный институт педагогических измерений”. *http://fipi.ru/ege-i-gve-11/demoversii-specifikacii-kodifikatory*

## Головоломки “Два квадрата” и “Три квадрата”

В октябрьском номере журнала за 2015 год, в разделе “В мир информатики”, было опубликовано следующее задание: “Можно ли нарисовать фигуры, изображенные справа, не отрывая карандаша от бумаги и не проводя дважды по одному и тому же отрезку? Если можно, то как?




Алгоритм решения задач/задачи, пожалуйста, оформите в виде:

1. AB.
2. BI.
3. …”

Тамара Николаевна Воронова, учитель информатики школы № 1 г. Рубцовска Алтайского края, сообщила в редакцию, что она использовала эти головоломки при проведении школьной олимпиады по информатике.

Заметим, что обе головоломки имеют несколько вариантов решения.

*Подготовил Д.М. Златопольский, редактор раздела “В мир информатики”*

ВСЕРОССИЙСКИЙ ПЕДАГОГИЧЕСКИЙ МАРАФОН УЧЕБНЫХ ПРЕДМЕТОВ

ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ г. МОСКВЫ
ИЗДАТЕЛЬСКИЙ ДОМ «ПЕРВОЕ СЕНТЯБРЯ»
МОСКОВСКИЙ ПЕДАГОГИЧЕСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ГЕНЕРАЛЬНЫЙ СПОНСОР: ИЗДАТЕЛЬСТВО «ДРОФА»

# 2016

22 МАРТА – 14 АПРЕЛЯ

## ВНИМАНИЕ!
## УТОЧНЕННОЕ РАСПИСАНИЕ ДНЕЙ ПЕДАГОГИЧЕСКОГО МАРАФОНА

| | |
|---|---|
| **22 марта** | **Открытие Марафона**<br>День классного руководителя<br>День учителя физической культуры |
| **23 марта** | День школьного психолога |
| **24 марта** | День здоровья детей, коррекционной педагогики, логопеда, инклюзивного образования и лечебной физической культуры |
| **25 марта** | День учителя начальной школы (день первый) |
| **26 марта** | День учителя начальной школы (день второй) |
| **27 марта** | День дошкольного образования |
| **28 марта** | День учителя технологии * |
| **29 марта** | День учителя географии |
| **30 марта** | День учителя химии |
| **31 марта** | День учителя биологии |
| **1 апреля** | День учителя информатики |
| **2 апреля** | День учителя физики |
| **3 апреля** | День учителя математики |
| **5 апреля** | День учителя истории и обществознания |
| **6 апреля** | День учителя МХК, музыки и ИЗО |
| **7 апреля** | День школьного и детского библиотекаря |
| **8 апреля** | День учителя литературы |
| **9 апреля** | День учителя русского языка |
| **10 апреля** | День учителя английского языка |
| **12 апреля** | День учителя французского языка |
| **13 апреля** | День школьной администрации |
| **14 апреля** | День учителя немецкого языка<br>**Закрытие** |

## marathon.1september.ru

! Обязательная предварительная регистрация на все дни Марафона с 22 февраля 2016 года на сайте marathon.1september.ru

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

В дни Марафона ведущие издательства страны представляют книги для учителей

Начало работы каждого дня – 9.00. Завершение работы – 15.00

## УЧАСТИЕ БЕСПЛАТНОЕ. ВХОД ПО БИЛЕТАМ

**РЕГИСТРИРУЙТЕСЬ, РАСПЕЧАТЫВАЙТЕ СВОЙ БИЛЕТ И ПРИХОДИТЕ!**

Место проведения Марафона: МПГУ, ул. Малая Пироговская, дом 1, стр. 1 (в 5 минутах ходьбы от ст. метро «Фрунзенская»)
* Место проведения Дня учителя технологии: ЦО № 293, ул. Касаткина, 1а (ст. метро «ВДНХ»)

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



## ИСТОРИЯ ИНФОРМАТИКИ

# “А зачем нужна эта ваша машина?”

В.В. Шилов,
Москва

Сегодня компьютеры (не только персоналки и ноутбуки, ведь любой современный бытовой прибор — телевизор, телефон, фотоаппарат и так далее, вплоть до холодильника и стиральной машины — содержит микропроцессор и фактически является компьютером!) окружают нас со всех сторон. Представить себе жизнь без компьютеров уже совершенно невозможно, поэтому нам трудно поверить, что семьдесят лет назад, когда первые электронные вычислительные машины (ЭВМ) только появились, кто-то мог не предвидеть их столь блестящее будущее.

Например, в прессе часто обсуждают слова, сказанные в конце 1940-х годов президентом компании IBM Томасом Ватсоном-старшим (*рис.* 1) о том, что на мировом рынке есть место не более чем для пяти ЭВМ. Звучит эта фраза действительно броско. Однако если разобраться, то окажется, что все обстоит отнюдь не так драматично. Ватсон был гениальным бизнесменом и организатором, за несколько десятилетий превратившим почти разорившуюся фирму в мирового лидера в области производства механических счетно-перфорационных

*Рис.* 1. Томас Ватсон

*Рис.* 2. Перфоратор IBM, 1935 г.

счетных машин и оборудования (табуляторы — см. *рис.* 2, перфораторы, сортировщики перфокарт и т.д.). Этот человек обладал феноменальным чутьем на все новое, сулящее успех, и при этом едва ли не лучше любого из современников понимал истинные потребности человечества в средствах вычислений. Но в то же время Ватсон мыслил вполне прагматично и в первую очередь учитывал интересы своего дела, а не развития абстрактных научных или технических идей.

В середине 1940-х гг. рыночные перспективы электронных компьютеров были в лучшем случае туманными. А компания IBM уже выпускала огромную номенклатуру электромеханических машин и оборудования для автоматизации вычислений. Ломать всю сложившуюся систему разработок и производства, бросаться с головой в омут новых и непроверенных электронных технологий было слишком рискованно. И в то время, когда многие разработчики и фирмы, сделавшие ставку на создание электронных машин, старались выпутаться из паутины технических и финансовых проблем, IBM, никуда не спеша, постепенно, шаг за шагом совершенствовала свои разработки, последовательно заменяя в вычислительных машинах механические и релейные узлы и устройства электронными. Архитектура машин совершенствовалась, но идеология обработки данных при этом не менялась, сохранялась преемственность и, самое главное, клиенты компании не теряли привычную для себя среду. И одновременно накапливался серьезный задел на будущее.

Правильность выбранной IBM стратегии обнаружилась достаточно быстро. Когда в начале 1950-х гг. в США стал формироваться компьютерный рынок, компания моментально вышла на него со своими новыми и оригинальными, но при этом уже обкатанными разработками и технологиями. Уже к концу десятилетия компания IBM настолько доминировала на этом рынке, что ее называли Белоснежкой — в противоположность “семи гномам” (под этим ироническим названием объединяли семь других компаний — разработчиков компьютеров).

*Рис.* 3. Говард Айкен

Столь же часто, как Ватсона, цитируют журналисты и другого человека, знаменитого компьютерного пионера, создателя вычислительной машины “Mark I” (1944 г.) Говарда Айкена. Из одной популярной статьи в другую кочует история о том, как он заявил в 1948 г., что для удовлетворения всех вычислительных потребностей Америке достаточно иметь пять машин. Эти слова приводят в качестве яркого примера свойственной иногда даже самым выдающимся ученым близорукости.

В действительности же все обстояло не совсем так. Как-то раз на одном из многочисленных в те годы совещаний рассматривались несколько проектов новых ЭВМ. Их архитектуры были достаточно похожи друг на друга, и Айкен высказал сомнение в целесообразности финансирования всех этих разработок. Его мысль заключалась в том, что надо сначала четко понять достоинства и недостатки уже построенных аналогичных машин (а их и было пять-шесть штук!), а уже потом расходовать огромные средства на новые. Кроме того, в словах Айкена был еще один нюанс, который сегодня может уловить только историк. На самом деле его крайне беспокоил вопрос, как подготовить достаточное число квалифицированных программистов, которые смогли бы обеспечить постоянную загрузку компьютеров. Он видел, что даже в то время, когда реально работали всего лишь две-три вычислительные машины, проблема программирования для них стояла крайне остро. Каждую ЭВМ обслуживали как минимум десяток людей, писавших программы. Если учесть, что программированием тогда занимались только математики высокой квалификации (которых никогда не бывает много!), то озабоченность Айкена, связанная с дальнейшим увеличением числа компьютеров, становится вполне понятной. (Разумеется, надо помнить, что программировали тогда в машинных кодах, и процесс отладки был крайне сложным и длительным. Никто, и Айкен здесь не исключение, не мог предположить, что в недалеком будущем появятся языки программирования высокого уровня и программирование из доступного только избранным искусства превратится в массовую профессию.)

Означают ли приведенные выше примеры, что на самом деле никакой недооценки возможностей компьютеров не было изначально, а их значение было сразу осознано? Разумеется, нет. Путь компьютеров к признанию был достаточно труден.

Выдающийся российский математик академик М.А. Лаврентьев в своих воспоминаниях рассказывает о том, какое сопротивление встречала идея автоматизации вычислений. Так, он пишет, как еще в 1935 году в Москве, в Математическом институте имени В.А. Стеклова была создана “вычислительная ячейка” (лаборатория), которая выполняла заказы на производство крупных расчетов. В ней работали математики (вычислители), вооруженные карандашом, бумагой и настольными арифмометрами (см. *рис.* 4). Постепенно заказов становилось все больше, и к 1947 году из двух комнат секция распространилась на целый этаж и занимала более половины всей площади института! Дальше расти было попросту некуда. И, кроме того, масштаб решаемых задач стал таким, что люди с ними уже не справлялись. (Стоит сказать, что такие же вы-

*Рис.* 4. Вычислители за работой. США, 1938 г.

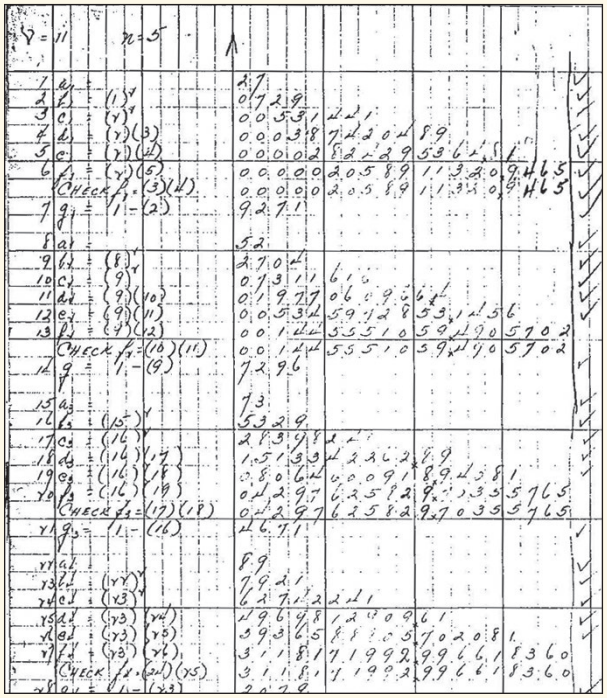


*Рис.* 5. Результаты работы вычислителей

числительные бюро, работавшие по заказам правительства и частных фирм, существовали во всех развитых странах — США, Великобритании и др.) Казалось бы, появление ЭВМ должно было обрадовать математиков, освобождаемых от рутинной и нудной работы. Однако на самом деле среди них произошел раскол. Большинство считали ЭВМ бесперспективной, сугубо рекламной затеей и предлагали не тратить на нее время и средства, а вместо этого увеличивать производство привычных аналоговых и механических счетных машин. Собственно говоря, знаменитый Институт точной механики и вычислительной техники (ИТМиВТ), в котором впоследствии разрабатывались все самые мощные советские суперкомпьютеры, был организован в 1948 году вовсе не для работ по созданию вычислительных машин! Основными направлениями его деятельности были определены разработка беспроводных систем управления, моделирование полета стратегических ракет, вычисление баллистических таблиц для средств ПВО, а также выполнение различных вычислительных работ по заказу Министерства обороны. Разработка вычислительных машин среди этих задач поначалу не значилась…

Так что первые проекты вычислительных машин советские разработчики пробивали с огромным трудом. Будущий академик Сергей Алексеевич Лебедев (на *рис*. 6 он — третий слева) начал обдумывать устройство цифровой электронной вычислительной машины еще в 1945 году. Спустя некоторое время В.А. Голубцова, ректор Московского энергетического института и жена всесильного тогда члена Политбюро ЦК КПСС Г.М. Маленкова, узнала о работе С.А. Лебедева и в беседе с ним пообещала организовать встречу с кем-либо из верхов. (Между прочим, затем она пригласила к себе на разговор жену Лебедева и поинтересовалась, имеется ли у Сергея Алексеевича подходящий к случаю костюм. Такового у ученого не оказалось, поэтому был выписан ордер в специальный магазин для покупки крайне дефицитной в то время ткани, и в специальном ателье костюм срочно пошили. В нем Лебедев и пошел на прием к члену ЦК, в ведении которого находилась советская наука.)

Лебедев подробно рассказал о том, что такое электронная вычислительная машина, что она сможет делать и во сколько обойдется ее постройка. Больше всего начальника заинтересовала скорость работы машины. Цифра 1000 операций в секунду произвела на него сильное впечатление. Начальник долго переваривал это фантастическое сообщение и затем изрек: “Значит, мы за пару месяцев перерешаем на вашей машине все свои задачи, а потом что — на помойку ее выбрасывать?” Возвращаться к разговору было бессмысленно. Вскоре С.А. Лебедев переехал из Москвы в Киев, где местная академия наук создала ему условия для работы, результатом которой стала первая отечественная ЭВМ — МЭСМ.

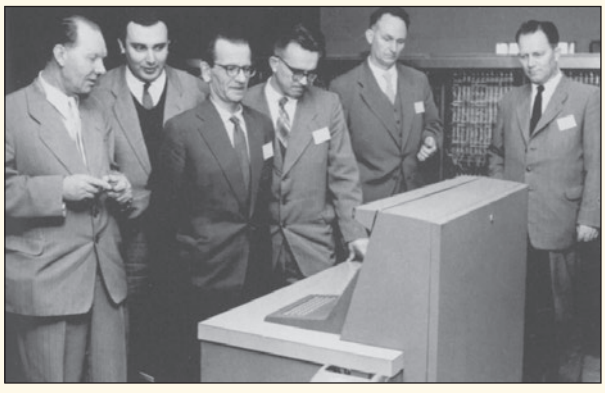


*Рис.* 6

Аналогичный случай в те же годы произошел еще с одним советским компьютерным пионером. Когда в кабинете большого чиновника от науки им были произнесены те же слова о возможностях электронных вычислительных машин, тот поинтересовался: “Так сколько, вы говорите, будет стоить эта ваша машина и сколько людей она сможет заменить?” Услышав, что стоимость машины может составить не один миллион рублей, чиновник (по совместительству также являвшийся ректором известного московского вуза) произнес: “Когда мне надо что-то посчитать, я сажаю на это дело сотню-другую студентов, и они все рассчитывают. Дешево и сердито”.

Даже в 1954 году, по воспоминаниям одного из создателей первой отечественной ЭВМ М-1[1] Башира Искандаровича Рамеева, академик М.В. Келдыш так говорил ему о машине “Стрела”: “Если бы таких ЭВМ выпустить 5–7 штук, то для Советского Союза этого было бы вполне достаточно”.

Впрочем, похожая ситуация была, например, и в Англии. Вскоре после Второй мировой войны здесь возникли несколько групп энтузиастов, которые вели соответствующие разработки, — не случайно первый в мире компьютер с хранимой в памяти программой EDSAC был построен именно в этой стране. Но очень многие научные эксперты сомневались в перспективности этого направления деятельности. Некоторые вообще полагали, что компьютер — это не более чем забавная игрушка для ученых. Правда, находить средства на экспериментальные образцы все-таки удавалось. Однако попытки вывести компьютер на рынок встречали полное непонимание даже у специалистов.

Один из британских компьютерных пионеров, профессор Дуглас Р. Хартри, в сентябре 1951 г. заявил представителям фирмы “Ferranti”, искавшим покупателя на изготовленную ею вычислительную машину: “Компьютер есть в Кембридже; компьютер есть в Манчестере и еще один в Национальной физической лаборатории. Я думаю, что нужен еще один в Шотландии, но это и все”. И Хартри не был тогда одинок в отрицательном отношении к коммерциализации компьютерной отрасли. Например, гениальный математик Джон фон Нейман, внесший колоссальный вклад в теорию и практику вычислительной техники, резко возражал против намерений создателей первого электронного ком-

[1] М-1 и МЭСМ начали работать в декабре 1951 года практически одновременно, с разницей в несколько дней. Поэтому вряд ли стоит отдавать пальму первенства одной из них; обе по справедливости могут быть названы “первой отечественной ЭВМ”.

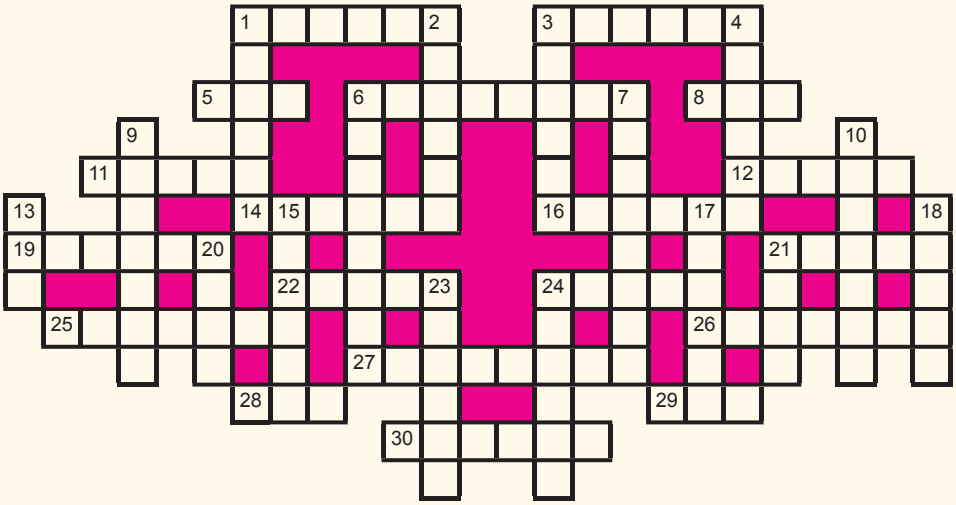
пьютера ENIAC Джона Моучли и Преспера Эккерта организовать массовое производство компьютеров. Фон Нейман полагал, что это “непрофессионально”!

Что же, мы видим, что многие выдающиеся специалисты действительно не сразу и далеко не в полной мере оценили возможности электронных компьютеров. Но, упрекая их в близорукости, не выступаем ли мы, по выражению Бориса Пастернака, в роли “пророка, предсказывающего назад”? Тем более, как писал Сергей Есенин, “большое видится на расстояньи”… И уверены ли мы, что наше понимание значения и возможностей научных открытий сегодняшнего дня когда-нибудь не покажется таким же недальновидным и наивным?

## “ЛОМАЕМ” ГОЛОВУ

### Кроссворд

Решите, пожалуйста, кроссворд.




*По горизонтали*

1. Условный знак или физический процесс, передающие некоторую информацию.

3. Оболочка одного из основных элементов настольного персонального компьютера.

5. Буква латинского алфавита, которой, как правило, обозначают неизвестную величину.

6. Совокупность четко определенных правил для решения задачи за определенное число шагов.

11. ru, ua, us, com, info и т.п.

12. Операция, проводимая с переменными *x* и *y* в фрагменте программы:

```
всп := x
x := y
y := всп
```

14. Примечание к тексту, размещаемое в нижней части страницы или в конце документа.

16. Так называют информацию, которая передается по Интернету.

19. Смещение первого символа первой строки абзаца.

21. Цирковая … .

22. Определение, нахождение объекта (например, элемента массива) по имени или другому признаку.

24. Основной структурный элемент презентации, созданной с помощью программы Microsoft PowerPoint.

25. Кривая, отдельные части которой построены с использованием рекурсивного подобия (то есть которые аналогичны основной кривой, но имеющие другие характеристики).

26. Цифра пятеричной системы счисления.

27. Участник почтовой переписки (отправитель).

28. Основание системы счисления, используемой в компьютере.

29. … симметрии.

*По вертикали*

1. Тригонометрическая функция, обозначаемая *sec*.

2. Наука о законах и формах мышления.

3. Программа, обращающаяся с запросами к другой программе (почти антоним к слову *сервер*).

4. Перечень предметов, людей и т.п.

6. Наука, изучающая звук.

7. Единица измерения информации, равная 1024 Кб.

9. Один из первых языков программирования высокого уровня.

10. Арифметическая операция.

13. Число в системе условных обозначений символов.

15. Нарост (на стволе или корне дерева, на копытах или т.п.), а также скопление, появление людей в большом количестве.

17. Номер элемента массива.

18. Линия связи.

20. Цифра восьмеричной системы счисления.

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

23. Шрифт наклонного начертания.

24. Устройство для ввода информации в персональный компьютер.

### Термины по одной теме

В приведенной ниже таблице вписаны 10 терминов, связанных с одной темой. Термины могут быть “изломаны” под прямым углом сколько угодно раз

и читаться во всех направлениях. Найдите эти термины по комментариям к ним и установите тему, с которой они связаны.

1. Наука о способах и формах мышления.
2. Форма мышления, с помощью которой из одного или нескольких суждений (посылок) получается новое (вывод).
3. “1” в логике.
4. Логическое отрицание.
5. Форма мышления, фиксирующая признаки объекта.
6. Логическое умножение.
7. Форма мышления, в которой что-либо утверждается или отрицается о предметах.
8. Логическое сложение.
9. “0” в логике.
10. Логическое следование.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| С | У | У | М | О | З | Я | И | С |
| Е | Ж | Д | Л | К | А | В | Е | Р |
| И | Н | Е | Ю | Ч | И | Н | П | О |
| Л | Е | И | Н | Е | И | Т | Я | Н |
| О | Г | И | К | А | Е | Ъ | Ю | Н |
| И | С | Т | И | Д | И | З | Ц | К |
| О | К | А | Н | Ж | О | Л | И | Я |
| Н | Н | К | Я | Ь | Л | И | Ц | И |
| Ъ | Ю | Ц | И/И | М | П | К | А | Я |

## Числовой ребус “Лестница из слов”

Решите, пожалуйста, числовой ребус:

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | | Д |
| | | | | | А | Д |
| + | | | Л | А | Д | |
| | | К | Л | А | Д | |
| | О | К | Л | А | Д | |
| Д | О | К | Л | А | Д | |
| 9 | 8 | 7 | 6 | 5 | 4 | |

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

## Ребусы в четверичной системе счисления

Здесь также одинаковые цифры зашифрованы одинаковыми буквами, разные цифры — разными буквами, но зашифрованы числа в четверичной системе счисления.

1.
| | | | |
|---|---|---|---|
| | X | X | |
| + | | X | |
| | X | Z | |

2.
| | | | |
|---|---|---|---|
| | E | E | |
| + | | E | |
| | D | F | |

3.
| | | | |
|---|---|---|---|
| | A | A | |
| + | | A | |
| | B | C | D |

Ответы (можно не ко всем ребусам) присылайте в редакцию.

## Потерянная гиря

Было 8 гирь весом 1 кг, 2 кг, …, 8 кг без надписей. Все гири выполнены из одного материала, так что чем больше вес гири, тем больше ее размер (но при этом размер не пропорционален весу). Одну из гирь потеряли. Необходимо за два взвешивания на чашечных весах выяснить, какая именно гиря потеряна.

## Переложить одну спичку[3]

Как, переложив одну спичку, получить верное равенство? Найдите два решения.


# ЛИЧНОСТИ

## Крис Касперски

*Когда произносят фамилию “Касперский”, большинство опытных (и не очень) пользователей компьютеров вспоминают известного российского программиста, автора распространенной антивирусной программы Евгения Касперского. Однако есть еще один, менее известный в широких кругах пользователей, программист, связанный с указанной фамилией. Правда, “Касперски” (именно так!) — это его псевдоним. Об этом человеке мы и хотим рассказать*[2].

Крис Касперски (настоящее имя Николай Владимирович Лихачев) родился в 1976 году в селе Успенское Краснодарского края. В 7 лет собрал свой первый работающий радиоприемник. Первый компьютер — “Правец 8D”[4] — ему привез из командировки отец. Компьютер подключался к магнитофону — аудиокассеты заменяли ему жесткий диск, а в качестве монитора выступал телевизор. Инструкция к компьютеру была на болгарском языке, но у знакомых смогли найти словарь.

На “Правец 8D” будущий Крис Касперски написал свою первую программу — простенькую игру на тему рыбалки. Нолик и знак “>” (0>) символизировали рыбку, которая плавала по экрану взад-вперед, а в центре был рыбак в виде знака вопроса. Чтобы поймать рыбку, нужно было нажимать на клавишу “Пробел”.

Следующий компьютер был “Электроника БК 0010”, на котором Крис освоил язык программи-

[2] Статья подготовлена по материалам сайта *polit.ru*.

[3] Задание предназначено для учащихся начальной школы и учеников 5–7-х классов.

[4] “Правец 8D” — персональный компьютер, выпускавшийся в Болгарии.

рования ассемблер. Потом были “ZX Spectrum” и “Агат”.

В школе Крис учился хорошо, закончил ее на одни пятерки. Без экзаменов поступил в Таганрогский радиотехнический университет на специальность “Проектирование микроконтроллеров”. Но, проучившись всего три месяца, бросил учебу и вернулся в родное село, поскольку в институте “программировать толком не давали”.

“Год я сидел дома и ничего не делал, потом снова поступил, чтобы успокоить маму, и снова бросил. Но тогда мне купили IBM, жесткий диск на 20 мегабайт и цветной монитор, целых шестнадцать цветов”, — рассказывает Крис.

Во время учебы в Таганроге он познакомился с предприимчивым студентом Шуриком, с которым они на паях открыли бизнес — оказывали услуги системного администрирования. Однако через некоторое время компаньон скрылся с деньгами, а Крису пришлось расплачиваться с рэкетирами. Отдав оставшиеся деньги и компьютер, он снова вернулся в село к родителям.

Все дальнейшие попытки организовать свой компьютерный бизнес в Армавире, Краснодаре и Ростове-на-Дону также не увенчались успехом.

С 1990 года Крис активно писал в так называемых “эхоконференциях” Фидо[5]. В частности, написал целую серию книг и статей, содержащих “рабочие” техники работы с дизассемблером и отладчиком.

В 1999 году знакомый свел Криса с издательством технической литературы “Солон Пресс”, где тиражом десять тысяч экземпляров вышла его первая книга “Техника и философия хакерских атак”. На момент написания книги ему было всего 22 года.

Следующие десять лет Касперски вел затворническую жизнь сельского чудака, занимаясь написанием книг и статей для множества журналов, таких, как “Системный администратор”, “Байт”, “Хакер”, и др.

К 2008 году было издано уже шестнадцать книг Криса Касперски на русском и английском языках — по защите информации и оптимизации программ, компьютерным вирусам, дизассемблированию.

Примерно в это время в жизни Касперски неожиданно случился, как он сам пишет, “взрывной каскад путешествий по всему миру” по приглашениям на различные конференции. Всего за несколько лет он посетил огромное количество стран, в том числе и экзотических. Вершиной этого стало громкое выступление на тему удаленного взлома процессора Intel на конференции в Малайзии. Эти поездки дали Касперски возможность завести важные профессиональные знакомства и заключить несколько значимых контрактов по всему миру.

В июне 2008 года у Касперски появилась наконец постоянная работа. Американская фирма “Endeavor Security”, занимающаяся безопасностью компьютеров и сетей, взяла его независимым консультантом. В 2009 году “Endeavor Security” была приобретена компанией McAfee. В McAfee Касперски проработал до сентября 2012 г. В настоящее время работает экспертом по безопасности в фирме “Check Point Software Technologies”.

С 2008 года живет в США в “Мекке” программистов — городе Рестон и является обладателем редкой американской визы O1 для людей с выдающимися способностями. По некоторым данным, уже стал долларовым миллионером.

## Чем знаменит

Крис Касперски — один из самых известных в России специалистов в области компьютерной безопасности. Из-под его пера вышли такие книги, как “Техника и философия хакерских атак”, “Образ мышления дизассемблера IDA”, “Техника сетевых атак”, и многие другие книги и статьи по программированию, взлому и отладке программ.

Иногда в книжных аннотациях можно встретить фразы типа: “Имя Криса Касперски известно едва ли не каждому пользователю персонального компьютера России и за рубежом благодаря популярному антивирусному программному обеспечению”. Однако Крис не имел и не имеет никакого отношения к “Лаборатории Касперского” и не является родственником его основателя и создателя известного антивируса Евгения Касперского. В свое время, во избежание путаницы с Евгением Касперским, он даже удалил последнюю букву из своего псевдонима.

Его книги, помимо собственно технических аспектов, не лишены и весьма серьезной философской составляющей. Стиль Касперски сложно с кем-то спутать; в своих работах он не просто учит хакерским премудростям и рассказывает, как нужно взламывать и как защищаться от атак, но и задается вопросом — зачем это делать.

В 2008 году Крис Касперски обнаружил уязвимость в процессорах Intel, которая позволяет совершить удаленный взлом системы при помощи скрипта на JavaScript или TCP/IP-пакета вне зависимости от операционной системы. О существовании “дыр” в процессорах Intel специалисты догадывались давно, но Касперски первым доказал, что уже написаны программы, которые помогут пробираться через них в компьютер. “Я всего лишь доказал, что это не миф”, — говорит сам Касперски.

## О чем надо знать

Помимо компьютеров, еще одним огромным увлечением Криса Касперски является астрономия. Еще в детстве отец построил для него обсерваторию рядом с домом. “Помимо возни с железом, я не расстаюсь с миром звезд и моих телескопов”, — говорит сам Крис. В одном из своих интервью он даже заявил однажды, что не хотел бы жить в городе как раз потому, что там совсем не видно звезд из-за сильной засветки.

Известные астрономические сайты публикуют его статьи, посвященные отечественным телеско-

[5] Фидо́ (полное название — Фидоне́т (англ. FidoNet, от созвучного греческого слова, означающего “коротко”) — международная любительская компьютерная сеть, популярная в начале 1990-х годов (в бывшем СССР — до конца 1990-х).

пам и наблюдению звездного неба. Даже самая первая из опубликованных Крисом статей была посвящена астрономии и вышла в журнале “Звездочет” в 1995 году, когда он еще учился в школе.

В 2003 году вышла в свет книга Криса Касперски “Энциклопедия примет погоды. Предсказание погоды по местным признакам”.

## Прямая речь

*О себе:* “Небрежно одетый мыщъх, не обращающий внимание ни на мир, ни на тело, в котором живет, и обитающий исключительно в дебрях машинных кодов и зарослях технических спецификаций”.

*Об отношениях с Евгением Касперским:* “Я не знаю почему, но как только в лаборатории разговор заходит обо мне, у них сразу портится настроение. Теперь они распространяют слухи, что я у них три недели работал, они взяли меня смеха ради, а потом выгнали. Сейчас мы с Касперским в параллельных плоскостях находимся, я не мешаю его бизнесу, а он не мешает моему. Я даже специально убрал «й» из псевдонима, чтобы нас не путали. Но он меня все равно не любит”.

*О своих пристрастиях:* “Скорее я хиппи. Не вандализм, а исследование. Хакером теперь называют даже школьников. А я люблю слово «кодокопатель». Тот, кто роет вглубь”.

*О деньгах*: “Миллион — это очень мало на самом деле. Это верхне-средний класс. С другой стороны, одни интересные люди предлагали мне два «лимона». Я спросил у них, что можно сегодня купить на эти деньги. Интересные люди неправильно меня поняли и, не торгуясь, предложили сразу двадцать, но были посланы, потому что на текущие расходы я и сам заработаю, а большего мне не надо”.

*О вирусных угрозах*: “Самый последний писк моды — в прицел атаки попали встраиваемые устройства. В первую очередь это, конечно, роутеры. Зловредный код в роутере очень сложно обнаружить. А тем временем хакеры нашли способ проникнуть внутрь камер наблюдения, подключенных к Ethernet. На очереди “умная” бытовая техника (например, холодильники), а также атаки на бортовой компьютер автомобиля — это фантастика новой реальности”.

*Игорь Садреев о Крисе Касперски:* “Он говорит с заметным южным акцентом, иногда проглатывает целые предложения, коверкает слова и смягчает окончания глаголов. Маленького роста, худой — кожа да кости в данном случае не преувеличение”.

*Сотрудник фирмы “Endeavor” Элис Чанг об известности Криса Касперски:* “Известность Касперски за пределами России не преувеличена. Он очень известный хакер, причем именно в первоначальном значении этого слова: человек, который понимает самые основы того, как работает программа”.

## Восемь фактов о Крисе Касперски

1. Когда Крису (точнее, тогда — Николаю) было всего несколько недель, врач по ошибке сделал мальчику инъекцию хлористого кальция и малыш

перенес инсульт. Частично омертвели ткани мозга, что привело к легкому аутизму.

2. Псевдоним, из-за которого Криса постоянно путают с создателем антивирусных программ, появился от большой любви к мультфильму про привидение Каспера и к музыканту и композитору Крису Кельми: “Я никогда не слушал его музыку, но от самого фонетического сочетания можно с ума сойти”.

3. За исключением своего дня рождения, Крис вообще не помнит дат, а хронологию событий восстанавливает по моделям компьютеров, которые у него были в разное время.

4. Любит произведения Фрэнка Герберта и часто использует их для эпиграфов. Также читает Виктора Пелевина, Юлию Остапенко и Ф.Ю. Зигеля, о которых сам говорит так: “Именно они, как никто другой, повлияли на мое творчество, сформировали стиль и манеру изложения, зажгли искру, определили жизнь и заставили взяться за перо”.

5. Другой псевдоним — “мыщъх” — был придуман Крисом на основе романа “Дюна”, главный герой которого носил имя Муад’Диб, что означает пустынную мышь. Почти все остальные ники Криса — это то же слово “мышь” на разных языках мира. Иногда специально чуть искаженные, например: souriz — от *souris* (франц.).

6. Первое собеседование Криса в США закончилось неожиданно — он отказался выполнять предложенное задание, найдя в нем ошибки и “нечеткую спецификацию”. В итоге это собеседование окончилось предложением работы с окладом $300 000 в год — первой официальной работой в его жизни.

7. В первые годы жизни Касперски в Рестоне (США) к нему неожиданно нагрянуло с обыском ФБР, изъявшее все записи и носители. Никаких обвинений, однако, ему не предъявляли, а сама история закончилась внесудебным урегулированием с выплатой Крису “отступных в размере зарплаты сферического программиста из РФ за несколько лет”. Однако свои записи и наработки Касперски пришлось довольно долго потом восстанавливать. С тех пор самые важные данные он хранит на винчестере в банковской ячейке.

8. В США Крис всерьез увлекся коллекционированием стрелкового оружия и спортивной стрельбой.

[7] А также в системе малых ЭВМ (СМ ЭВМ). — **Прим. ред.**

## КРЕПКИЙ ОРЕШЕК

В этой рубрике, как обычно, мы проводим разбор задач, решение которых вызвало трудности.

### Задача “Необычная фигура на шахматной доске”

Напомним условие.

В левом нижнем углу шахматной доски[6] (см. *рис.* 1) стоит шахматная фигура, которую назовем “полукороль”, так как она, в отличие от “настоящего” шахматного короля, ходит только на одно поле строго вправо, или строго вверх, или строго по диагонали вправо-вверх. Сколько существует способов перемещения фигуры на поле, выделенное желтым цветом?

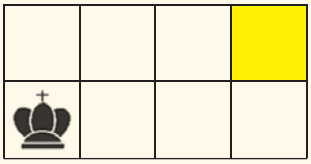


*Рис.* 1

Один из способов решения задачи заключается в анализе маршрутов, так сказать, “с конца”.

С полей, выделенных на *рис.* 2 синим цветом, до “цели” полукороль может переместиться единственным способом. Запишем это на рисунке:

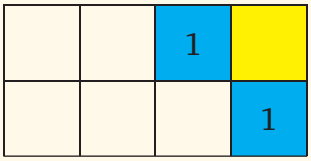



*Рис.* 2

С красных полей путь также единственный:

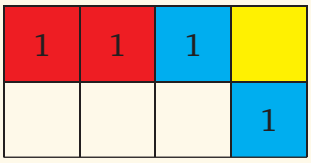



*Рис.* 3

Обсудим теперь число маршрутов с зеленого поля (*рис.* 4). С него на желтое поле можно попасть за один ход или через синие поля. Так как число маршрутов с синих полей нам уже известно (оно записано в этих полях), то общее число путей с зеленого поля до цели равно 1 (непосредственно) + 1 (через верхнее синее поле) + 1 (через правое синее поле) = 3. Запишем результат:

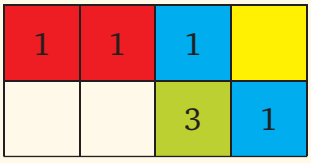



*Рис.* 4

[6] На всех рисунках статьи поля доски черного цвета условно не выделены.

Продолжая аналогичные рассуждения и записывая рассчитываемые значения, можно получить следующую картину:

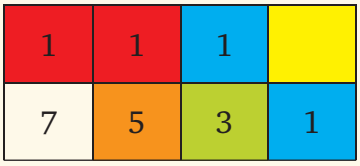



*Рис.* 5

Из *рис.* 5 видно, что искомое число маршрутов равно 7.

Для самостоятельной работы была предложена такая задача, связанная с “полукоролем”: “В правом нижнем углу фрагмента шахматной доски (см. *рис.* 6) стоит шахматная фигура, которую назовем “полукороль”, так как она, в отличие от “настоящего” короля, ходит только на одно поле строго влево, или строго вверх, или строго по диагонали влево-вверх. Сколько существует способов перемещения фигуры на поле, выделенное желтым цветом?

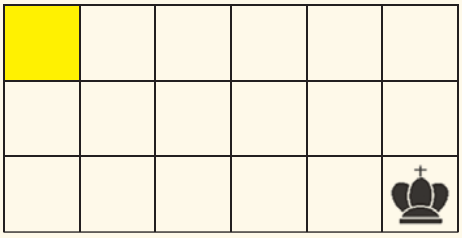


*Рис.* 6

На основе рассуждений, аналогичных сделанным выше, можно получить ряд значений (см. *рис.* 7).

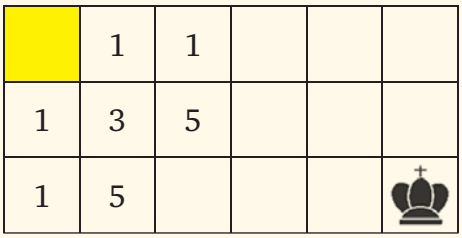



*Рис.* 7

Продолжив рассуждения, получите искомое число способов перемещения фигуры.

### Задание “Число кратчайших путей. Задача 1”

Напомним условие.

На поле шахматной доски, выделенном зеленым цветом (см. рисунок ниже), стоит “настоящий” шахматный король, то есть фигура, которая, в отличие от “полукороля” из предыдущей задачи, ходит по правилам шахмат. Сколько существует маршрутов перемещения короля на желтое поле доски за *наименьшее* число ходов?

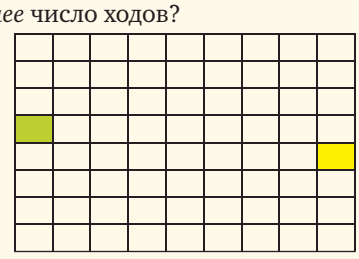


*Рис.* 1

*Решение*

Сначала удобно получить число шагов с того или иного поля доски до цели (поля *h*4). При этом также целесообразно начинать анализ “с конца”:

| | a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|---|
| 8 | | 6 | 5 | 4 | 4 | 4 | 4 | 4 |
| 7 | | 6 | 5 | 4 | 3 | 3 | 3 | 3 |
| 6 | | 6 | 5 | 4 | 3 | 2 | 2 | 2 |
| 5 | | 6 | 5 | 4 | 3 | 2 | 1 | 1 |
| 4 | | 6 | 5 | 4 | 3 | 2 | 1 | |
| 3 | | 6 | 5 | 4 | 3 | 2 | 1 | 1 |
| 2 | | 6 | 5 | 4 | 3 | 2 | 2 | 2 |
| 1 | | 6 | 5 | 4 | 3 | 3 | 3 | 3 |

*Рис.* 2

После этого можно начать подсчет количества маршрутов с того или иного поля на желтое поле доски за наименьшее число ходов.

Для белых полей в правой части доски (см. *рис.* 3) это количество равно 1:

| | a | b | c | d | e | f | g | h |
|---|---|---|---|---|---|---|---|---|
| 8 | | | | | | | | |
| 7 | | | | | 1 | 3 | 5 | 4 |
| 6 | | | | | 3 | 1 | 2 | 2 |
| 5 | | | | | 6 | 2 | 1 | 1 |
| 4 | | | | | 7 | 3 | 1 | |
| 3 | | | | | 6 | 2 | 1 | 1 |
| 2 | | | | | 3 | 1 | 2 | 2 |
| 1 | | | | | 1 | 4 | 5 | 4 |

*Рис.* 3

Определим искомое количество для полей серого цвета. С полей *g*6 и *h*6 до цели кратчайшим путем (за два хода) можно попасть двумя способами (по двум маршрутам) — через поля *g*5 и *h*5. Для полей *f*5 и *f*3 число маршрутов минимальной длины также равно двум. А вот с полей *f*2 и *f*6 до поля желтого цвета есть только один кратчайший маршрут — через поля *g*3 и *g*5, соответственно. Почему для поля искомое количество равно трем, подумайте…

Продолжая “пятиться” (по клеткам таблицы-“доски”, а не по комнате ☺), найдите все другие значения вплоть до “главного” искомого значения — для поля *a*5. При этом постарайтесь установить правило, по которому можно рассчитывать эти значения для полей того или иного цвета на *рис.* 2 и *рис.* 3.

## Задание “Число кратчайших путей. Задача 2”

Для каждого поля шахматной доски (см. рисунок ниже) определите число *кратчайших* маршрутов перемещения шахматного короля с этого поля на поле *a*8 (выделенное желтым цветом). Например, для полей *a*7, *b*8 и *b*7 это число равно 1.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | | | | | | |
| 1 | 1 | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |
| | | | | | | | |

Ответ оформите в виде таблицы 8 × 8.

Дополнительный вопрос: “С какого поля (с каких полей) число кратчайших маршрутов до цели наибольшее?”.

Задача решается аналогично предыдущей.

# ЯПОНСКИЙ УГОЛОК

## Два судоку

友

Решите, пожалуйста, две японские головоломки “судоку”:

1) простую:

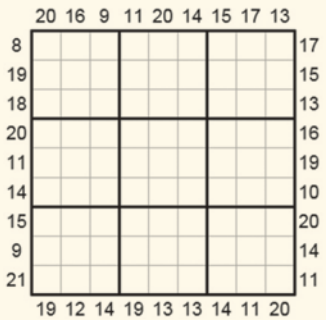
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 4 | | 2 | | 3 | 9 | 8 | | |
| | 6 | 9 | 5 | | 4 | | 3 | 2 |
| 7 | | | | 2 | | 1 | 4 | |
| | 7 | | | | | 4 | 9 | |
| | | 4 | 9 | | 1 | 2 | | |
| | 9 | 5 | | | | | 7 | |
| | 4 | 6 | | 9 | | | | 7 |
| 1 | 3 | | 6 | | 8 | 9 | 2 | |
| | | 8 | 3 |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

2) сложную:

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

## И судоку-рама

Этот тип японских головоломок — новый на страницах нашего журнала. Слово *рама* в его названии не связано с русским словом такого же написания. Этот вариант судоку вместо обычного задания содержит цифры вокруг сетки. Каждое из чисел обозначает сумму ближайших трех цифр в строке или в столбце. Как и в обычном судоку, задача состоит в том, чтобы вписать в ячейки числа от 1 до 9 так, чтобы в каждом горизонтальном ряду, в каждой вертикальной колонке и в каждом блоке каждое число встречалось только однажды.




Ответы присылайте в редакцию (можно решить отдельные головоломки).

# Ответы, решения, разъяснения

к заданиям, опубликованным в сентябрьском выпуске “В мир информатики” (окончание)

## Задача “Ли Ван и Ван Ли”

Напомним, что было предложено решить следующую задачу: “В некоторой стране, жители которой говорят или только правду, или только лгут, имеются два семейства — Ли Ван и Ван Ли, отличающиеся от обычных жителей. Семейство Ли Ван говорит правду только утром и лжет вечером, а семейство Ван Ли — правду только вечером, а ложь — утром.

Вы встретили двух человек — **A** и **B**, по одному из каждого семейства, но не знаете, кто из них к какому семейству принадлежит. Каждый сделал по одному утверждению:

**A**: “Сейчас вечер”;

**B**: “Моя фамилия Ван Ли”.

Какие семейства представляет каждый из встреченных?”.

*Решение*

Можно сделать те или иные допущения.

Допустим, что встреча состоялась вечером. Тогда утверждающий это **A** сказал правду, а по вечерам правдиво отвечает человек из семейства Ван Ли. Но тогда ответ **B** — ложный, то есть **B** — Ли Ван, который лжет по вечерам. Противоречий нет.

Если же встреча состоялась утром, то солгавший **A** — это Ван Ли. Из этого следует, что **B** — лжет. Но по утрам лжет Ван Ли, которым **B** быть не может. Имеет место противоречие.

Можно также рассуждать так. И вечером, и утром будет отвечать: “Сейчас вечер” человек из семейства Ван Ли (убедитесь в этом самостоятельно). Значит, ответивший так **A** и есть Ван Ли. Но тогда **B** — Ли Ван — лжет, то есть встреча состоялась вечером. Противоречий нет.

*Ответ*: **A** — из семейства Ван Ли, **B** — Ли Ван.

*Ответы прислали*:

— Антипенко Кирилл, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Бумажкин Сергей, Ростовская обл., г. Новочеркасск, Новочеркасское Суворовское училище МВД РФ, преподаватель **Воронкова О.Б.**;

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

— Дощик Константин и Назаркина Татьяна, средняя школа г. Пионерский Калининградской обл., учитель **Багрова О.А.**;

— Лежнева Александра, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Прокопенко Сергей, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

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

Все перечисленные читатели будут награждены дипломами. Поздравляем!

## Новая задача “Еще раз семейства Ли Ван и Ван Ли”

В некоторой стране, жители которой говорят или только правду, или только лгут, имеются два семейства — Ли Ван и Ван Ли, отличающиеся от обычных жителей. Семейство Ли Ван говорит правду только утром и лжет вечером, а семейство Ван Ли — правду только вечером, а ложь — утром.

Допустим, вы встретились с четырьмя представителями этих семейств (назовем их **A**, **B**, **C** и **D**), по два человека от каждого. Вот что сказал каждый встреченный:

**A**: “**B** — Ли Ван”;

**B**: “**C** — один из Ван Ли”;

**C**: “**A** и **D** — из различных семейств”;

**D**: “**A** и **B** — из одного и того же семейства”.

Когда могла произойти встреча — утром или вечером? Какие семейства представляет каждый из говоривших?

| № | Вариант | Высказывания | Анализ |
|---|---|---|---|
| 1 | **A** и **B** — Ли Ван | **A**: **B** — Ли Ван | Соответствие |
| | | **B**: **C** — один из Ван Ли | То же |
| | **C** и **D** — Ван Ли | **C**: НЕ(**A** и **D** — из различных семейств) | Противоречие (**A** и **D** — из различных семейств) |
| | | **D**: НЕ(**A** и **B** — из одного и того же семейства) | Противоречие (**A** и **B** — из одного семейства) |
| 2 | **A** и **B** — Ван Ли | | |
| | | | |
| | **C** и **D** — Ли Ван | | |
| | | | |
| 3 | **A** и **C** — Ли Ван | | |
| | | | |
| | **B** и **D** — Ван Ли | | |
| | | | |
| 4 | **A** и **C** — Ван Ли | | |
| | | | |
| | **B** и **D** — Ли Ван | | |
| | | | |
| 5 | **A** и **D** — Ли Ван | | |
| | | | |
| | **B** и **C** — Ван Ли | | |
| | | | |
| 6 | **A** и **D** — Ван Ли | | |
| | | | |
| | **B** и **C** — Ли Ван | | |
| | | | |

*Указания по выполнению*

Рассмотрите два возможных случая:

1) встреча произошла утром;

2) и встреча произошла вечером

— и для каждого из них — все возможные варианты сочетания фамилий пар людей (обозначив, например, представителей семейства Ван Ли — **A** и **B**, семейства Ли Ван — **C** и **D**). Для анализа удобно использовать такую таблицу:

**1. Первый случай — встреча произошла утром (Ли Ван говорят правду, Ван Ли — лгут)**

См. табл. на с. 59.

**2. Второй случай — встреча произошла вечером (Ван Ли говорят правду, Ли Ван — лгут)**

| № | Вариант | Высказывания | Анализ |
|---|---|---|---|
| 1 | **A** и **B** — Ли Ван | **A**: **B** — Ли Ван | |
| | | **B**: **C** — один из Ван Ли | |
| | **C** и **D** — Ван Ли | **C**: НЕ(**A** и **D** — из различных семейств) | |
| | | **D**: НЕ(**A** и **B** — из одного и того же семейства) | |
| … | | | |

## Задача “Сколько единиц и сколько нулей?”

Напомним, что следовало найти количество единиц $e$ и количество нулей $n$ в двоичной записи выражения:

$$(2^6 - 1) - (2^2 - 1).$$

Заметим, что в ряде ответов результат получен путем расчета заданного числа, перевода его в двоичную систему счисления и последующим подсчетом числа нулей и единиц. Конечно, такой способ решения возможен. Однако учитывая, что аналогичные задания представлены в Едином государственном экзамене по информатике, время проведения которого ограничено, опишем методику более быстрого получения результата.

Анализ таблицы:

| $k$ | $2^k$ | Двоичное представление числа $2^k$ | Двоичное представление числа $2^k - 1$ |
|---|---|---|---|
| 1 | 2 | 10 | 1 |
| 2 | 4 | 100 | 11 |
| 3 | 8 | 1000 | 111 |
| 4 | 16 | 10000 | 1111 |
| 5 | … | | |

— показывает, что $(2^k - 1)$ — это $k$-значное двоичное число, состоящее из одних единиц. Значит, разность двух чисел $(2^6 - 1)$ и $(2^2 - 1)$ будет состоять из двух нулей и четырех $(6 - 2)$ единиц.

*Ответ*: $e = 4$, $n = 2$.

*Ответы представили*:

— Антипенко Кирилл, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Благинина Александра, г. Ростов-на-Дону, лицей № 56, учитель **Назаренко С.Н.**;

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

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

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

— Неустроев Владислав, г. Пенза, школа № 512, учитель **Гаврилова М.И.**

## “Сколько единиц и сколько нулей?”. Новые задачи

Найдите количество единиц $e$ и количество нулей $n$ в двоичной записи выражения:

1) $(2^{115} - 1) - (2^{60} - 1)$;

2) $(2^a - 1) - (2^b - 1)$ при $a > b$.

Правильное решение задач, опубликованных в сентябрьском выпуске “В мир информатики”, представили также:

1) задачи “Работники кафе”:

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

— Рустамов Артем, г. Ярославль, гимназия № 1, учитель **Тишининова Я.Н.**;

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

— Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**;

2) задачи “Кто на каком инструменте играет?”:

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

— Назаров Петр, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

— Якушов Александр, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина, учитель **Чапкевич И.М.**;

3) задачи “Переправа”:

— Антипенко Кирилл, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

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

(В их решении больше “операций” по переправе, чем описано в решении, опубликованном в предыдущем выпуске “В мир информатики”.)

Решение задачи “Постройка плотины” представила также Маслова Арина, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**

Кроссворд, опубликованный в сентябре, решили также:

— Бабайцева Дарья, Бородулина Елизавета, Гатауллина Елизавета, Кочеткова Мария и Попов Илья, Рязанская обл., Сапожковская средняя школа имени Героя России А.И. Тучина, учитель **Задубровский В.Ф.**;

— Занозин Федор, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Пашкова Елизавета, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**, а японские головоломки судоку:

— Багрова Анастасия, Ветрова Таисия, Кукушкина Карина, Кутовая Мария и Шикалович Ростислав, средняя школа г. Пионерский Калининградской обл., учитель **Багрова О.А.**;

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

— Маслова Арина, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**

## Пятерка в конце

В каких системах счисления запись десятичного числа 27 оканчивается на 5?

## Гостиницы

Четыре друга — Владимир, Андрей, Александр и Михаил решили отдохнуть на одном из зарубежных курортов. Там все они поселились в разных гостиницах: “Тропикаль”, “Конкордия”, “Флорида” и “Леонардо”.

Известно, что:

1) Владимир проводил своих друзей в гостиницы “Леонардо” и “Тропикаль”;

2) тот из друзей, кто поселился в “Леонардо”, — дальний родственник Андрея;

3) житель гостиницы “Флорида” встречается с Владимиром и Михаилом у моря;

4) Андрей приходит обедать в гостиницу “Флорида” к своему товарищу.

Кто в какой гостинице живет?

## Последняя цифра

Найдите последнюю цифру числа $2^{50}$.

## Умная бабушка

Одна бабушка спросила каждого из своих семи внуков: “Какой сегодня день недели?”. Они ответили так:

1) первый внук: “Сегодня не воскресенье, а завтра не среда”;

2) второй внук: “Вчера была не пятница, а позавчера был не понедельник”;

3) третий внук: “Завтра не воскресенье, и вчера было не воскресенье”;

4) четвертый внук: “Послезавтра не суббота и не воскресенье”;

5) пятый внук: “Вчера был не понедельник и не среда”;

6) шестой внук: “Позавчера была не среда, а завтра не вторник”;

7) седьмой внук: “Сегодня не среда”.

Дедушка сказал, что ответ одного из внуков — ложный, после чего бабушка, конечно, подумав немного, определила, какой же день недели был на самом деле (а внуку, сказавшему неправду, сделала замечание).

А сможете ли определить “настоящий” день недели вы, уважаемый читатель?

## Улов

Рыбак поймал окуня, ерша и щуку. Щуку он поймал раньше, чем окуня, а ерша — позже, чем щуку. Можно ли сказать, какая рыба была поймана раньше: окунь или ерш?

Задача предназначена для учащихся 1–7-х классов.

## Волшебные цветные рыбки

В аквариуме плавают 2 красные, 20 синих и 5 зеленых рыбок. Если друг друга касаются две рыбки одинакового цвета, то ничего не происходит, а если же разного цвета, то тогда они принимают третий цвет. В каком случае все рыбки могут стать одного цвета? После какого минимального числа касаний это может произойти? Какого цвета будут рыбки?

Ответ оформите в виде:

| Этапы | Встречаются | | Будет две новых | Станет | | |
|---|---|---|---|---|---|---|
| | | | | Красных | Синих | Зеленых |
| Исходное состояние | | | | 2 | 20 | 5 |
| 1 | Красная | Синяя | Зеленые | 1 | 19 | 7 |
| … | | | | | | |

## Пятеро друзей

В поездке пятеро друзей — Антон, Борис, Вадим, Дима и Гриша, познакомились с попутчицей. Они предложили ей отгадать их фамилии, причем каждый из них высказал одно истинное и одно ложное утверждение.

Дима сказал: “Моя фамилия — Мишин, а фамилия Бориса — Хохлов”. Антон сказал: “Мишин — это моя фамилия, а фамилия Вадима — Белкин”. Борис сказал: “Фамилия Вадима — Тихонов, а моя фамилия — Мишин”. Вадим сказал: “Моя фамилия — Белкин, а фамилия Гриши — Чехов”. Гриша сказал: “Да, моя фамилия Чехов, а фамилия Антона — Тихонов”.

Какую фамилию носит каждый из друзей?

# ПОИСК ИНФОРМАЦИИ

## Четыре вопроса

1. Кем в фильме “Срочно в номер” приходилась героиня, роль которой играла Валентина Талызина, герою фильма, роль которого исполнял Ярослав Бойко?

2. Кого Ханс Кристиан Андерсен советует остерегаться в своей сказке “Нехороший мальчик”?

3. На гербе какой страны изображен голубь?

4. Датские рыбаки, знаменитые на весь мир своими рекордными уловами, в большинстве своем даже не знают вкуса рыбы, а отдают предпочтение другим продуктам. Среди них много бобовых и круп. Но первое место по популярности занимает совсем уже не морской продукт. Какой?

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

## Числовой ребус с “МАССИВОМ”

Напомним, что следовало решить числовой ребус:

$$\mathbf{МА}^{\mathbf{С}} = \mathbf{СИВ},$$

в котором, как принято в таких головоломках, одинаковыми буквами зашифрованы одинаковые цифры, разными буквами — разные цифры.

Благодаря Елизавету Пашкову, Республика Карелия, поселок Надвоицы, школа № 1 (учитель **Каликина Т.В.**), Елизавету Удалову, средняя школа села Сердар, Республика Марий Эл (учитель **Чернова Л.И.**), и Александра Якушова, г. Орел, лицей № 4 им. Героя Советского Союза Г.Б. Злотина (учитель **Чапкевич И.М.**), приславших ответы, приведем начало решения.

Цифра **С** не может быть равна трем, так как даже небольшое двухзначное число 12 в кубе — это четырехзначное число. Значит, **С** = 2. С учетом этого ребус принимает вид:

$$\mathbf{РО}^{2} = 2\mathbf{ИЯ}$$

Осталось проанализировать числа третьей сотни, являющиеся точным квадратом. Предлагаем читателям провести такой анализ и прислать ответ в редакцию. Решите, пожалуйста, еще один числовой ребус со словом-числом **МАССИВ**:

$$\mathbf{МАС} = \mathbf{СИ}^{\mathbf{В}}$$

Ответ также присылайте в редакцию.

## Числовой ребус “АБВГД-йка”

Напомним, что ребус имел вид:

$$\begin{array}{r} \mathbf{АБВГД} \\ + \quad \mathbf{АБВГД} \\ \mathbf{АБВГД} \\ \mathbf{АБВГД} \\ \hline \mathbf{ДГВБА} \end{array}$$

Прежде всего видно, что **А** = 1 или **А** = 2. Но нет такой цифры **Д**, для которой число 4 × Д оканчивалось бы на цифру 1. Значит, **А** = 2, а это возможно при **Д** = 3 или при **Д** = 8. Анализ двух этих вариантов предлагаем провести читателям и прислать ответ в редакцию. А здесь мы отмечаем читателей, представивших правильное решение и которых редакция решила наградить дипломами:

— Валиеву Лилиану, Васвиеву Альбину и Садриева Дениса, средняя школа села Сейтяково Балтачевского р-на, Республика Башкортостан, учитель **Загафуранова А.Ф.**;

— Назаркину Татьяну, средняя школа г. Пионерский Калининградской обл., учитель **Багрова О.А.**

# ЦИФРОВОЙ МИР

## Искусственный интеллект учится мыслить логически

15 октября российские разработчики представили новую возможность искусственного интеллекта — способность к чтению текста.

Новинку представил основатель так называемого “миварного подхода” в области создания искусственного интеллекта Олег Варламов (мивар — многомерная информационная варьирующаяся адаптивная реальность).

— Обучаем мы его совсем как ребенка, сначала даем базовые знания, затем на них накладываем дополнительные и постепенно усложняем технологию, делая ее все более интеллектуальной, — отметил Олег. — В наших планах через год создать полностью самостоятельного робота, который бы общался с аудиторией без постороннего вмешательства.

Ну а пока интеллекту загружают текст примерно такого уровня: “На ферме живет много разных животных. Среди них свиньи, коровы, козы”. На вопрос: “Какие животные живут на ферме?”, машина, пользуясь специальной текстовой схемой, четко выдает: “Свиньи, коровы и козы”, и это не просто поиск по соответствиям: система идентифицирует местоимения, сложносочиненные и сложноподчиненные предложения.

— Очень важно, что мы развиваем технологию логического искусственного интеллекта, и обучаем мы ее на русском языке, — подчеркнул Олег Варламов. — Скоро эти технологии можно будет использовать в повседневной жизни: консультации по продуктам и услугам, общение с посетителями сайта, ответы на вопросы клиентов по услугам компании.

В одном из крупнейших российских банков в настоящее время тестируется виртуальный сотрудник службы поддержки, созданный на миварной основе. Другие возможные области применения технологии — автоматизированное осмысленное реферирование текстовых массивов, анализ законодательных актов на непротиворечивость и даже основа систем поиска плагиата в научных работах.

*По материалам газеты “Вечерняя Москва”,*
*автор публикации — Сергей Рожков*

# ВНИМАНИЕ! КОНКУРС!

## Конкурс № 116

В качестве задания этого конкурса предлагаем решить логическую задачу “Еще раз семейства Ли Ван и Ван Ли” (см. рубрику “Задачник”).

Ответ отправьте в редакцию по адресу: 121165, Москва, ул. Киевская, д. 24, “Первое сентября”, “Информатика” или по электронной почте: *vmi@1september.ru*. Срок представления ответов не ограничен.

Общероссийский проект

# Школа цифрового века

Министерство образования Московской области ● Издательский дом «ПЕРВОЕ СЕНТЯБРЯ»

# Электронные учебники — каждому ученику!

В 2016 году Министерство образования Московской области совместно с Издательским домом «Первое сентября» в рамках контракта с длинным, но значимым для современной школы названием: **«Оказание услуги по обеспечению доступа обучающихся общеобразовательных организаций Московской области к электронным учебникам и электронным приложениям к учебникам»** реализует проект по предоставлению современных электронных учебников всем ученикам и учителям Московской области.

В течение 2016 года педагоги и ученики 5–11-х классов Московской области будут обеспечены (за счёт бюджета Московской области) электронными учебниками ведущих издательств по всем предметам школьной программы.

Это первый для нашей страны проект такого масштаба по предоставлению электронных учебников всем ученикам региона.

Рано или поздно этот опыт будет подхвачен и другими регионами.

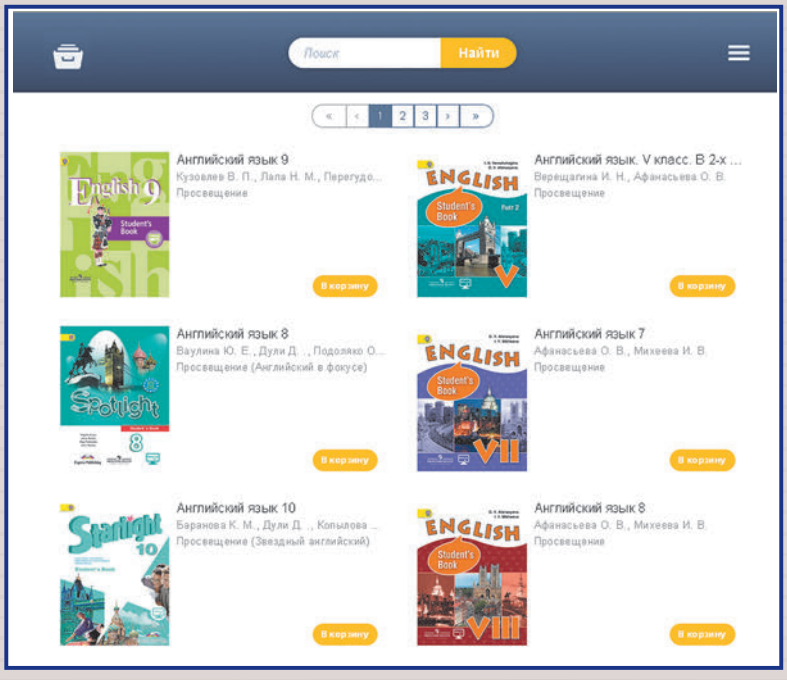
## Следите за проектом!

Наверняка ваш регион планирует переход на ЭУ в будущем!

## Включайтесь в проект,

если вы работаете в школе Московской области!

Витрина проекта: ibook.1september.ru




Подробности на сайте digital.1september.ru в разделе «Электронные учебники»

# НОВЫЙ ПРОЕКТ

«Первого сентября»



СПЕЦИАЛИСТЫ-ПРАКТИКИ

СВИДЕТЕЛЬСТВО УЧАСТНИКА

О ВОСПИТАНИИ

ОБ ОТНОШЕНИЯХ

О РОДИТЕЛЬСКОЙ ПОЗИЦИИ

О САМООЦЕНКЕ

АБОНЕМЕНТЫ

ОБ ОПТИМИЗМЕ

УДОБНОЕ

О ЦЕЛЯХ

О ЧУВСТВЕ ВИНЫ

О РАБОТЕ

О МИРОВОСПРИЯТИИ

О КОММУНИКАЦИИ

ВРЕМЯ

О ДЕТЯХ

# ВЕБИНАРЫ

О ДЕТЯХ С ОВЗ

ВОСТРЕБОВАННЫЕ

О КОНФЛИКТАХ

О СЕМЬЕ

ВИДЕОЗАПИСИ

ТЕМЫ

ДОСТУПНАЯ СТОИМОСТЬ

О МЕТОДАХ ОБУЧЕНИЯ

О ВЫГОРАНИИ

О КАРЬЕРЕ

О ВЗАИМОПОНИМАНИИ

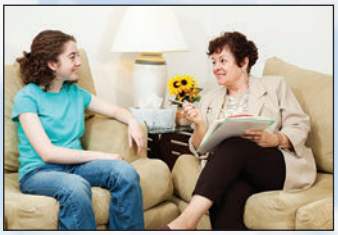
О СТРЕССЕ

ВОПРОСЫ И ОТВЕТЫ ОНЛАЙН

О ЦЕННОСТЯХ

О ЛИЧНЫХ КРИЗИСАХ





Видеозаписи вебинаров на сайте

**webinar.1september.ru**