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

ИНФОРМАТИКА

31 января — день БЕЗ ИНТЕРНЕТА

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

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

НА ОБЛОЖКЕ

У нас есть целый месяц для того, чтобы спланировать, как мы проведем БЕЗ Интернета 31 января. В этот день Интернет будет отключен по всему миру. Шутка :). Отказ от Интернета в последнее воскресенье января — дело сугубо добровольное, но следуют ему все большее количество людей по всему миру.

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

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 14.10.2015,
фактически 14.10.2015
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

Интернет изменил нашу жизнь. Как и насколько — темы многочисленных исследований. В любом случае каждый из нас на своем личном опыте знает, что и коммуникации, и поиск информации “до” и сейчас изменились принципиально. Время не повернуть вспять — как бы кому не нравилось то, что вместо читальных залов библиотек мы предпочитаем поисковую строку и экран смартфона, это реальность, созданная Интернетом. И, кстати, постепенно становится понятно, что не заменил и не заменит он всё и вся. Кинематограф не “убил” театр, а телевидение не “уничтожило” кинематограф с театром вместе взятые. Мир переформатировался и нашел новый баланс. То же происходит и с Интернетом.

Но хоть раз в году ведь можно позволить себе поностальгировать! Тем более что в какой именно день, известно. Ежегодно в последнее воскресенье января отмечается Международный день без Интернета, учрежденный неизвестно кем и неизвестно когда (Гугл нам в помощь… хотя, нет, в свете импортозамещения, конечно, лучше пользоваться Яндексом…). Поисковики расскажут нам, что свою историю этот праздник ведет с начала 2000-х годов, а инициатором его учреждения, по одним данным, является “Британский институт социальных изобретений”, а по другим — британский некоммерческий онлайн-проект “DoBe.org”. Единственное, что известно точно, — организаторами Международного дня без Интернета стали именно активные, продвинутые интернет-пользователи.

Ежегодные мероприятия, проходящие в различных странах мира в день без Интернета, свидетельствуют о том, что тема с годами приобретает все большую актуальность. Организаторы данных акций подчеркивают, что они не ведут борьбу с Всемирной сетью и не считают Интернет чем-то плохим. Они просто хотят вновь обратить внимание людей, и в первую очередь молодежи, что и в реальном мире много всего интересного и увлекательного. И что хотя бы один день можно провести с любимыми и близкими людьми, посвятить его своему хобби, сходить в музей или на выставку, или даже просто погулять в лесу… Международный день без Интернета имеет приверженцев во многих городах мира, в том числе и в России. Кстати, в нашей стране в этот день не остаются в стороне и учреждения образования, например, библиотеки, которые в честь этого праздника приглашают к себе в гости на различные культурные и просветительские мероприятия.

В общем, получив этот номер в начале января, у вас есть достаточно времени, чтобы спланировать для себя (а может быть, и вместе со своими учениками), как вы проведете 31 января — именно на последний день месяца выпадает этот интересный праздник в наступившем году.

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

Практикум программирования на ActionScript

УРОК 9. СОЗДАНИЕ ТЕСТОВ

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

Специальные имена и элементы пути

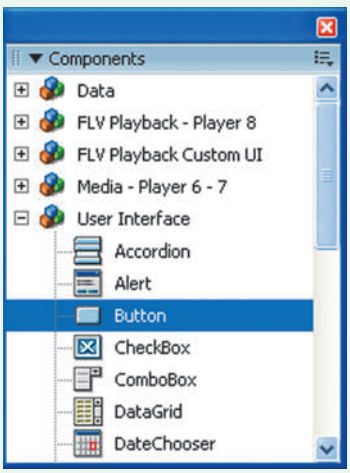
`this` — указатель на сам текущий объект, используется при вызове свойства или метода текущего объекта;

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

Добавление диалоговых элементов (“компонентов”)

“Компоненты” (элементы диалога) — это специальные стандартные объекты, которые содержатся в отдельной библиотеке. Их образцы (прототипы) имеются в ОС Windows, а пользователь может лишь выбирать и вставлять в кадр экземпляры объектов из этой библиотеки.

Библиотека элементов диалога раскрывается в отдельном окне выбором в главном меню команды **Window → Components**. В от-



Продолжение. Начало см. в № 12/2015.

крывшемся окне библиотеки компонентов нужно выбрать раздел **User Interface** (“интерфейс пользователя”).

Чтобы добавить на кадр экземпляр объекта, достаточно перетащить его из соответствующей строки библиотеки компонентов на кадр.

Основные свойства объектов — диалоговых элементов

Объект	Описание объекта	Свойство	Описание свойства
Button	Кнопка	label	Текст надписи на кнопке
RadioButton	Переключатель	data	Строка текста, привязанная к переключателю
		groupName	Имя группы переключателей. Переключатели всегда объединяют в группу; из всей группы может быть выбран только один переключатель
		label	Надпись рядом с переключателем
		selected	Переключатель выбран (true) или не выбран (false)
CheckBox	Флажок	label	Надпись рядом с флажком
		selected	Переключатель выбран (true) или не выбран (false). Флажки в группы не объединяются и обрабатываются каждый по отдельности
TextInput	Поле ввода текста	text	Текст в поле ввода (может быть задан изначально путем присваивания нужной строки свойству text или получен из поля ввода чтением этого свойства в переменную)
		editable	Возможность изменения содержимого поля (true/false)
		password	Скрывать ли текст в поле заменой символов на “звездочки” или точки, как в полях ввода паролей (true/false)

_visible — видимость объекта (**true** — видимый, **false** — невидимый).

Размерами прямоугольной области диалогового элемента (содержащей надпись рядом с переключателем или флажком) либо размерами поля ввода можно управлять через свойства ширины и высоты, как для любых других объектов Flash.

Глобальные переменные

Переменные, объявленные внутри обработчика события, присвоенного какому-либо объекту, используются в пределах этого объекта и этого обработчика. Точно такая же (одноименная) переменная в другом обработчике другого объекта — это уже совершенно другая переменная, и компьютер их не путает. Такие переменные, действующие в пределах “своего” объекта и обработчика, называются ***локальными***.

Переменная, которую можно использовать во всех объектах всех кадров анимации, называется ***глобальной***. 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(enterFrame) { *программный код* } — обработчик событий, назначаемый символу-клипу или кнопке (символу либо диалоговому элементу Button). Здесь используется событие **enterFrame** — “запуск данного кадра на воспроизведение”.

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.

on (press) { *программный код* } — обработчик событий, назначаемый для объекта-мишени. Данное событие также является внешним и выполняется многократно, каждый раз при нажатии левой кнопки мыши на этом объекте.

Типовая логика построения тестов

Тест представляет собой одно или несколько заданий, где каждое задание размещается на отдельном ключевом кадре. При этом первому ключевому кадру присваивается команда **stop();** для остановки ани-

мации и возможности выполнить имеющееся на кадре задание. Дальнейшие переходы на последующие кадры выполняются при помощи отдельного объекта-символа “Кнопка” (нарисованного, а не выбранного из библиотеки элементов диалога!) скриптом, содержащим в обработчике ее нажатия **on (press)** команду **gotoAndStop(<номер следующего кадра>)**.

Для итоговой обработки результатов тестирования выделяется отдельный ключевой кадр в конце анимации. Все переменные, содержащие информацию для итоговой обработки (количество правильных ответов, количество неправильных ответов и пр.), должны быть объявлены как глобальные.

Задание с выбором одного ответа реализуется при помощи переключателей. На кадре размещаются текст и изображения, соответствующие вопросу. Создается несколько переключателей, соответствующих вариантам ответов (рядом с ними можно разместить и варианты ответа в виде картинок). Все эти переключатели нужно объединить в одну группу, задав для них одинаковое имя группы **groupName**. Дополнительно на кадре размещается кнопка (Button) для проверки ответа на это задание. Этой кнопке назначается обработчик события **on (press)**, в котором размещаются команды проверки ответа: оператор **if** содержит в своем условии обращение к свойству **selected** правильного переключателя, и, если он выбран (данное свойство равно **true**), в основной ветви увеличивается количество правильных ответов и выполняются другие действия, соответствующие правильному ответу. Иначе в ветви **else** выполняются действия, соответствующие неверному ответу. При необходимости кнопке Button может быть назначен обработчик события **onClipEvent(load)**, в котором могут быть произведены какие-то инициализирующие действия в пределах данного кадра.

Задание с выбором нескольких правильных ответов реализуется с помощью флажков. На кадре размещаются вопрос и один или несколько флажков, соответствующих вариантам ответов. Там же размещается кнопка проверки. В скрипте ее обработчика **on (press)** нужно отдельно проверять операторами **if** значение свойства **selected** для каждого флажка: оно должно быть равно **true** для всех правильных ответов и **false** — для всех неправильных. После проверки полученного сложного условия выполняются действия, соответствующие правильному выполнению данного задания либо неправильному его выполнению.

Задание с вводом ответа реализуется с помощью поля ввода. На кадре размещаются вопрос и поле ввода нужного размера, в которое пользователь вписывает текст своего ответа (поле ввода должно иметь достаточную длину). Свойство **editable** этого поля ввода должно быть равно **true** (можно менять содержимое поля ввода), а свойство **password** должно быть равно **false** (текст виден как есть). Там же размещается кнопка проверки. В скрипте ее обработчика **on (press)** нужно запрашивать и сверять с правильной строкой текста значение свойства **text** данного поля ввода. Далее выполняются действия, соответствующие правильному выполнению данного задания либо неправильному его выполнению.

Используя функции обработки текста, на последующих занятиях мы научимся выделять во введенной пользователем строке некие “ключевые” (важные) слова, что позволит обрабатывать ответ независимо от используемых в нем падежей, спряжений и пр., а также от наличия “незначащих” предлогов и пр.

В рассмотренных вариантах задания выполнены так, что можно многократно делать повторные попытки ответа и нажимать кнопку проверки. Чтобы дать возможность лишь однократного ответа (нажатия кнопки проверки), можно в конце скрипта обработчика нажатия этой кнопки проверки записать команду **this._visible=false;**, которая спрячет эту кнопку.

Отдельно в кадре предусматривается команда для перехода на следующий кадр. Для этого нужно в кадре создать отдельную нарисованную кнопку, преобразовать ее в объект-символ типа “Кнопка” (Button) и присвоить ему обработчик нажатия **on (press)** с командой **gotoAndStop(<номер следующего кадра>);**.

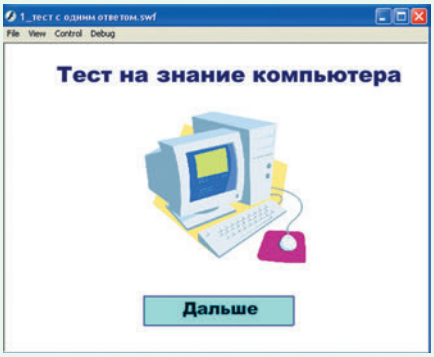
В последнем кадре размещается поле ввода (с значением свойства **editable**, равным **false**, — так что содержимое этого поля можно только читать, но не изменять) и кнопка “Результат”. Этой кнопке присваивается обработчик нажатия **on (press)**, в котором производится обращение к глобальной переменной с накопленными результатами. В зависимости от них в поле ввода выводится сообщение о результатах выполнения теста (присваивание нужной строки текста свойству **text** этого поля).

В профессионально создаваемых тестах всё и проще, и сложнее. В них флеш-анимация представляет собой лишь оболочку для отображения заданий теста и для обработки ответов и содержит только один кадр с заданием каждой разновидности. Дополнительно вместе с такой флеш-анимацией хранится файл собственно с заданиями (вопросами, вариантами ответов и указаниями, какие ответы правильные), в виде простого текста либо, чаще всего, в формате XML. При выполнении анимации из этого внешнего файла считывается сначала тип задания и в зависимости от него воспроизводится нужный кадр анимации. Далее из внешнего файла считываются текст вопроса и тексты вариантов ответа и выводятся в предусмотренных на кадре надписях. Варианты ответов при этом могут случайным образом перемешиваться и каждый раз выводиться в другом порядке. При ответе пользователя выполняется проверка данного задания и соответственно меняется значение глобальных переменных — счетчиков правильных и неправильных ответов. Далее из внешнего файла считывается следующее задание и т.д. После их исчерпания в анимации выводится итоговый кадр с результатами теста. Мы же пока будем создавать более простой вариант теста без использования внешнего файла.

Практические задания

Задание I. Тест с выбором одного правильного ответа

1. Создадим новую анимацию. В ее первом ключевом кадре разместим название теста и прочую “оформительскую” информацию (например, Ф.И.О. разработчика теста). Здесь же нарисуем кнопку “Дальше” и преобразуем ее в библиотечный символ-кнопку Button.



2. Выделив мышью первый ключевой кадр, раскроем окно редактора сценариев. Привяжем к этому кадру следующие команды:

```
p_otv=0;
n_otv=0;
stop();
```

```
Начальные присваивания:
- глобальная переменная — количество правильных ответов
- глобальная переменная — количество неправильных ответов
- остановка воспроизведения анимации на первом ее кадре
```

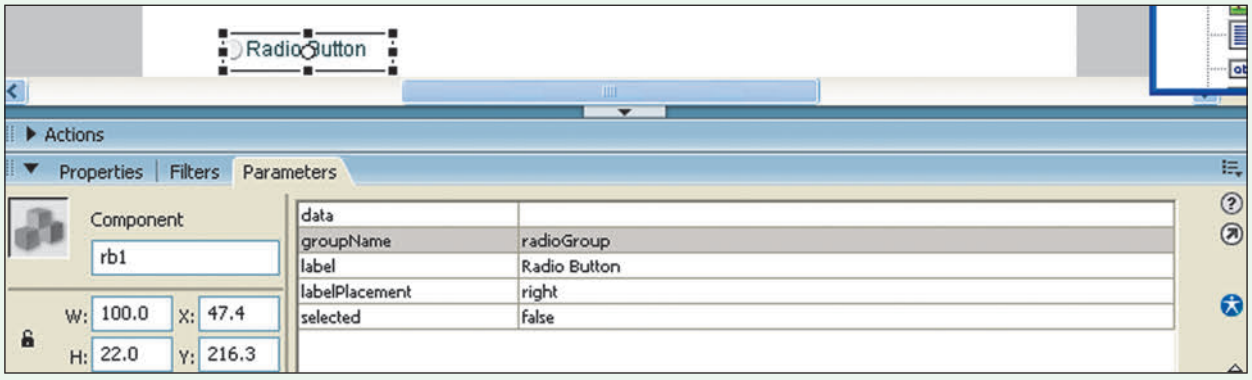
3. К кнопке “Дальше” привяжем сценарий обработчика нажатия:

```
on (press) {
   gotoAndStop(2);
}
```

```
Нажатие кнопки:
- перейти на кадр №2 и остановить воспроизведение
```

4. Добавим второй кадр как ключевой. Раскроем панель библиотеки компонентов (элементов диалога). Разместим на втором кадре текст вопроса для первого задания, например: “Какое устройство из минимального комплекта ПК является основным?”.

5. Перетаскивая из панели библиотеки компонентов образец **RadioButton**, разместим на кадре пять переключателей. Назначим им имена экземпляров, соответственно, **rb1**, **rb2**, **rb3**, **rb4**, **rb5**. Выделив переключатель, раскроем в панели параметров (внизу) вкладку **Parameters**.



В таблице зададим следующие значения свойств:

groupName = rbgroup (одно и то же значение названия группы для всех переключателей),

selected = false (одно и то же значение для всех переключателей — ни один из них изначально не помечен),

в свойстве **label** введем текст вариантов ответов — для каждого переключателя свой:

Переключатель (имя экземпляра)	Текст варианта ответа (свойство label)
rb1	монитор
rb2	системный блок
rb3	клавиатура
rb4	мышь
rb5	аудиоколонки

Для переключателей, надпись которых не отображается полностью, увеличим ширину — значение в поле **W** (в левой части панели параметров) увеличим так, чтобы надпись была видна вся.

6. Перетаскивая из панели библиотеки компонентов образец **Button**, разместим на кадре кнопку проверки. Для нее зададим имя экземпляра **butpr1** и значение свойства **label**, равное тексту “Проверка”.

7. Перетаскивая из панели библиотеки компонентов образец **TextInput**, разместим на кадре поле ввода текста, которое пока (временно) будет служить нам для вывода результата. Для него зададим имя экземпляра **rez1** и значение свойства **editable = false**. Значение свойства **text** оставим пустым.

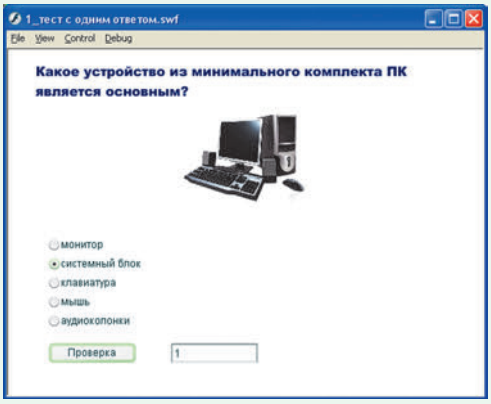
8. Созданной кнопке с надписью “Проверка” присвоим скрипт:

```
onClipEvent(load) {
 _root.rez1._visible = false;
}
on (press) {
  rbprav = _root.rb2.selected;

  if (rbprav) {
    _root.p_otv += 1;
    _root.rez1._visible = true;
    _root.rez1.text = _root.p_otv;
  } else {
    _root.n_otv += 1;
    _root.rez1._visible = true;
    _root.rez1.text = _root.p_otv;
  }
}
```

```
Начальные присваивания:
– изначально поле ввода скрыто

Нажатие кнопки:
– считываем, выбран ли второй
  переключатель (правильный)
– если true, то
  увеличиваем счетчик правильных ответов,
  показываем поле ввода,
  выводим в нем значение p_otv;
– иначе
  увеличиваем счетчик неправильных ответов,
  показываем поле ввода,
  выводим в нем значение p_otv
```

Так как в группе может быть выделен только один переключатель, для проверки правильности ответа достаточно проверить, помечен ли правильный переключатель.

9. Запустим анимацию на выполнение. Проверим: при выборе правильного ответа (“системный блок”) в поле ввода текста (которое появится после нажатия кнопки проверки) должно появиться число 1, а при вводе неправильного ответа — число 0.



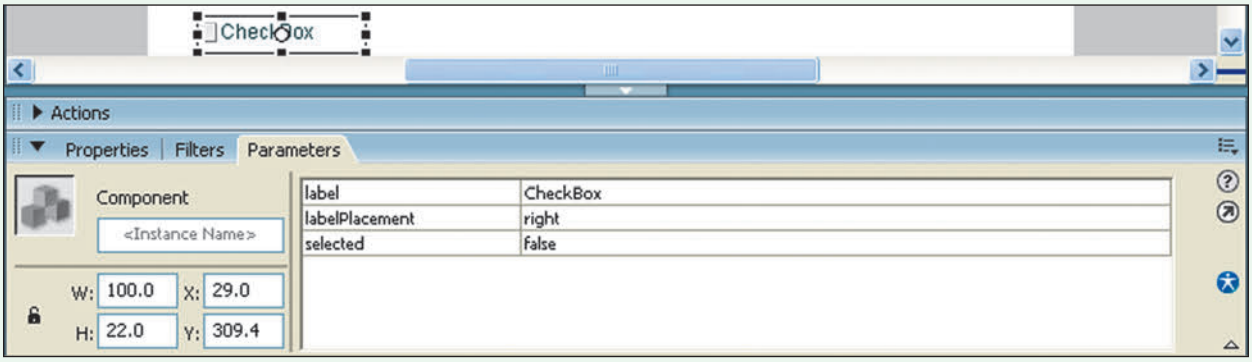
Задание II. Тест с выбором нескольких правильных ответов

10. Продолжим редактирование анимации, созданной в предыдущем задании. Создадим новый ключевой кадр в позиции №3 таймлайна.

Кстати, при этом нетрудно заметить, что на него автоматически будут скопированы все объекты с предыдущего, второго кадра. Если новое задание предполагается того же типа, что и предыдущее, то можно соответствующим образом отредактировать эти объекты. Иначе нужно удалить лишнее (в нашем случае — переключатели), оставив только те объекты, которые нам понадобятся (текст вопроса, кнопку проверки и поле ввода текста, в котором выводится результат).

11. Раскроем панель библиотеки компонентов (элементов диалога), если она еще не раскрыта. Разместим на третьем кадре текст вопроса для второго задания, например: “Выберите среди перечисленных периферийных устройств устройства вывода”.

12. Удалим с кадра все скопировавшиеся туда переключатели. Перетаскивая из панели библиотеки компонентов образец **CheckBox**, разместим на кадре пять флажков. Назначим им имена экземпляров, соответственно, **fl1**, **fl2**, **fl3**, **fl4**, **fl5**. Выделив флажок, раскроем в панели параметров (внизу) вкладку **Parameters**.



В таблице зададим следующие значения свойств:
`selected = false` (одно и то же значение для всех флажков — ни один из них изначально не помечен), в свойстве **`label`** введем текст вариантов ответов — для каждого переключателя свой:

Переключатель (имя экземпляра)	Текст варианта ответа (свойство label)
fl1	клавиатура
fl2	монитор
fl3	мышь
fl4	принтер
fl5	аудиоколонки

Для флажков, надпись которых не отображается полностью, увеличим ширину — значение в поле **W** увеличим так, чтобы надпись была видна вся.

13. Отредактируем объект **Button** (кнопку проверки, которая оказалась скопирована в кадр при его создании из предыдущего второго кадра). Для нее изменим имя экземпляра на **butpr2**.

14. Отредактируем объект **TextInput** (поле вывода текста, которое также оказалось скопировано в кадр при его создании из предыдущего второго кадра). Это поле временно будет служить нам для вывода результата выполнения данного задания. Для него изменим имя экземпляра на **rez2**, а значения свойств оставим те, которые были скопированы вместе с этим объектом.

15. Скрипт кнопки с надписью “Проверка”, который был скопирован вместе с ней из предыдущего кадра, изменим на следующий:

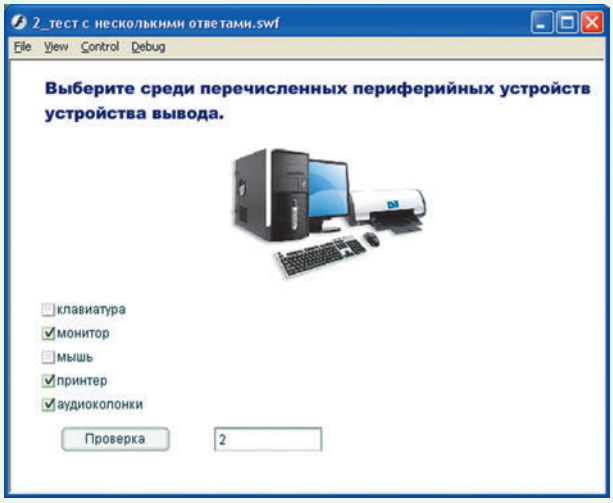
```
onClipEvent(load) {
 _root.rez2._visible = false;
}
on (press) {
  flag1 = _root.fl1.selected;
  flag2 = _root.fl2.selected;
  flag3 = _root.fl3.selected;
  flag4 = _root.fl4.selected;
  flag5 = _root.fl5.selected;
  prav = !flag1 && flag2 && !flag3
         && flag4 && flag5;
  if (prav) {
    _root.p_otv += 1;
    _root.rez2._visible = true;
    _root.rez2.text = _root.p_otv;
  } else {
    _root.n_otv += 1;
    _root.rez2._visible = true;
    _root.rez2.text = _root.p_otv;
  }
}
```

```
Начальные присваивания:
– изначально поле ввода скрыто

Нажатие кнопки:
– считываем по очереди все флажки




– формируем условие правильного ответа
  ("!" – операция "НЕ")
– если true, то
  увеличиваем счетчик правильных ответов,
  показываем поле ввода,
  выводим в нем значение p_otv;
– иначе
  увеличиваем счетчик неправильных ответов,
  показываем поле ввода,
  выводим в нем значение p_otv
```

В данном случае, в отличие от использования переключателей, ответ на задание будет правильным, только если помечены все “правильные” флажки и не помечен ни один “неправильный”. Поэтому нужно считывать состояние каждого имеющегося флажка, а затем формировать сложное условие, где операцией И (`&&`) объединены логические значения — считанные перед этим состояния всех флажков: для “правильных” флажков, которые должны быть помечены, записывается само их состояние, а для “неправильных”, которые не должны быть помечены, перед соответствующим логическим значением ставится знак “`!`” (логическая операция отрицания).

16. Скопируем кнопку “Дальше” с первого кадра на второй. В скрипте этой кнопки изменим в команде `gotoAndStop( );` в скобках номер кадра с 2 на 3.

17. Запустим анимацию на выполнение. Проверим: при пропуске первого задания (если сразу нажать кнопку “Дальше”) при правильном ответе на второе задание (“монитор”, “принтер”, “аудиоколонки”) в поле ввода текста, которое появится после нажатия кнопки проверки, должно появиться число 1, а при неправильном ответе на второе задание — число 0.

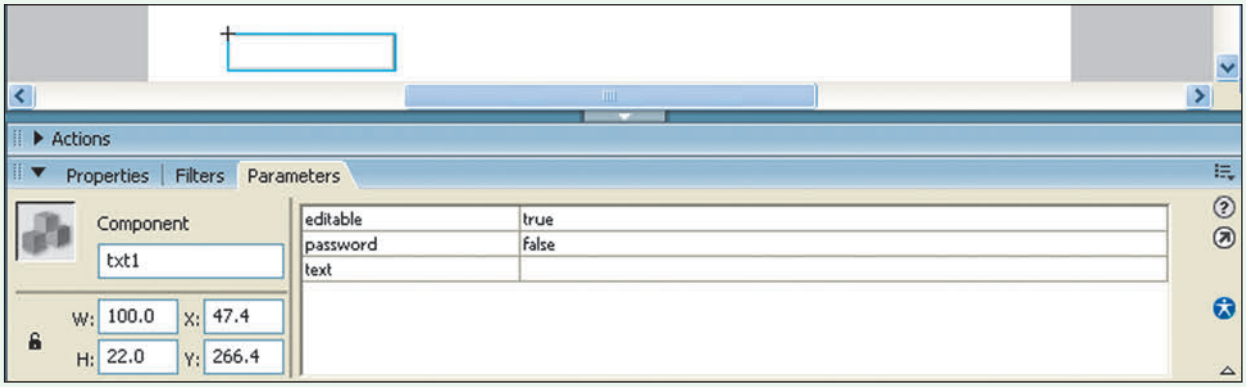


Задание III. Тест с вводом ответа в текстовой форме

18. Продолжим редактирование анимации, созданной в предыдущем задании. Создадим новый ключевой кадр в позиции №4 таймлайна.

19. Раскроем панель библиотеки компонентов (элементов диалога), если она еще не раскрыта. Разместим на четвертом кадре текст вопроса для третьего задания, например: “Какое устройство позволяет вводить в компьютер графическую информацию с бумажного источника?”.

20. Перетаскивая из панели библиотеки компонентов образец **TextInput**, разместим на кадре поле ввода текста. Назначим ему имя экземпляра **txt1**. Выделив это поле ввода, раскроем в панели параметров (внизу) вкладку **Parameters**.



В таблице зададим следующие значения свойств:

`editable = true` (проверяем, что это значение выставлено по умолчанию),

`password = false` (также должно стоять по умолчанию),

значение свойства **`text`** должно быть пустым.

Также увеличим размеры поля — значения в полях **W** и **H**.

21. Отредактируем объект **Button** (кнопку проверки), скопировав ее в данный кадр из второго кадра. Для нее изменим имя экземпляра на **butpr3**.

22. Отредактируем объект **TextInput** (поле вывода текста), который также скопируем из второго кадра. Это поле временно будет служить нам для вывода результата выполнения данного задания. Для него изменим имя экземпляра на **rez3**, а значения свойств оставим те, которые были скопированы вместе с этим объектом.

23. Скопируем кнопку “Проверка”. Скрипт, который был скопирован вместе с этой кнопкой, меняем на следующий:

```
onClipEvent(load) {
 _root.rez3._visible = false;
}
on (press) {
  totv = _root.txt1.text;

  prav = (totv == "сканер") ||
         (totv == "Сканер") ||
         (totv == "СКАНЕР");
  if (prav) {
    _root.p_otv += 1;
    _root.rez3._visible = true;
    _root.rez3.text = _root.p_otv;
  } else {
    _root.n_otv += 1;
    _root.rez3._visible = true;
    _root.rez3.text = _root.p_otv;
  }
}
```

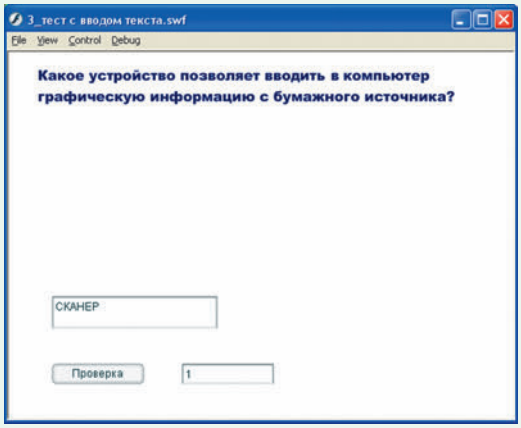
```
Начальные присваивания:
– изначально поле ввода скрыто

Нажатие кнопки:
– считываем текст, введенный в поле ввода
  ответа
– перебираем все возможные варианты
  ввода правильного ответа (используется
  логическая операция ИЛИ (|||))
– если true, то
  увеличиваем счетчик правильных ответов,
  показываем поле ввода,
  выводим в нем значение p_otv;
– иначе
  увеличиваем счетчик неправильных ответов,
  показываем поле ввода,
  выводим в нем значение p_otv
```

Пока мы только сравниваем весь текст, введенный пользователем в качестве ответа, с правильным названием устройства, которое может быть записано в трех вариантах — все строчные буквы, все заглавные или начиная с заглавной. (При записи этих образцов правильного ответа внимательно следите за регистром клавиатуры: ввод вместо русской буквы похожей на нее английской нарушит работу программы!) На последующих занятиях мы освоим работу с текстовыми данными и научимся обрабатывать введенный текст более корректно.

24. Скопируем кнопку “Дальше” с первого (или второго) кадра. В скрипте этой кнопки изменим в команде **gotoAndStop( );** в скобках номер кадра на 4.

Напомним: если при вставке ранее скопированного в буфер объекта дополнительно удерживать нажатой клавишу **Shift**, то объект вставится на то же место, как и на кадре, из которого он был скопирован.

25. Запустим анимацию на выполнение. Проверим: при пропуске первых двух заданий (если сразу нажимать кнопку “Дальше”) при правильном ответе на третье задание (“сканер”) в поле ввода текста (которое появится после нажатия кнопки проверки) должно появиться число 1, а при неправильном ответе на второе задание — число 0.



Задание IV. Создание итогового кадра с результатами тестирования

26. Продолжим редактирование анимации, созданной в предыдущем задании. Создадим новый ключевой кадр в позиции №5 таймлайна. Удалим все скопировавшиеся в этот кадр элементы.

27. Создадим на кадре необходимое оформление, например:

28. Перетаскивая из панели библиотеки компонентов образец **TextInput**, разместим на кадре три поля ввода текста напротив каждой из надписей с двоеточиями. Назначим им имена экземпляров (сверху вниз): **sumkol**, **pravotv** и **nepravotv**. В панели параметров на вкладке **Parameters** для всех трех полей ввода установим значения свойств:

editable = false (запрет изменения содержимого),
password = false (должно стоять по умолчанию),
значение свойства **text** должно быть пустым.

29. Верхнему полю ввода (экземпляр **sumkol**) присвоим скрипт:

```
onClipEvent(enterFrame) {
  this.text = _root.p_otv + _root.n_otv;
}
```

```
Действия при открытии кадра:
— общее число заданий можно определить
  как сумму правильных и неправильных
  ответов
```

Среднему полю ввода (экземпляр **pravotv**) присвоим скрипт:

```
onClipEvent(enterFrame) {
  this.text = _root.p_otv;
}
```

```
Действия при открытии кадра:
— вывести кол-во правильных ответов
```

Нижнему полю ввода (экземпляр **nepravotv**) присвоим скрипт:

```
onClipEvent(enterFrame) {
  this.text = _root.n_otv;
}
```

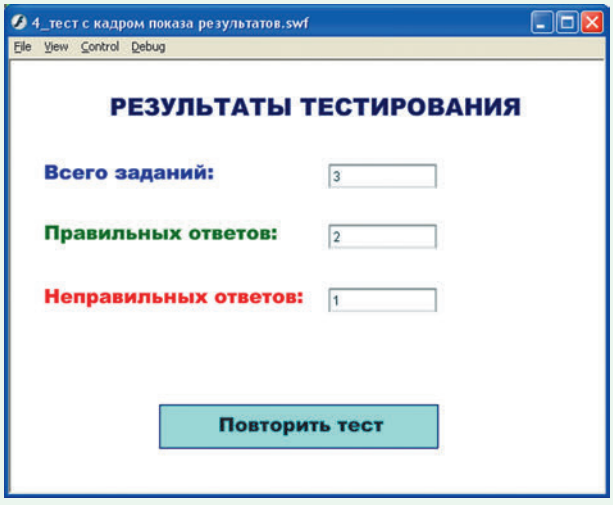
```
Действия при открытии кадра:
— вывести кол-во неправильных ответов
```

30. Скопируем кнопку “Дальше” с любого из первых кадров на четвертый. В скрипте этой кнопки изменим в команде **gotoAndStop( );** в скобках номер кадра на 5.

31. На последнем, пятом кадре добавим нарисованную кнопку (аналогичную кнопке “Дальше”) с надписью “Повторить тест”. Преобразуем ее в символ типа **Button**. Присвоим ей скрипт:

```
on (press) {
  _root.p_otv = 0;
  _root.n_otv = 0;
  gotoAndStop(2);
}
```

32. Запустим анимацию на выполнение. Выполним имеющиеся тесты и проверим вывод результатов. Рекомендуется выполнить проверку анимации несколько раз, нажимая кнопку “Повторить тест”, и давать как правильные, так и неправильные ответы, чтобы проверить все возможные варианты.



Задание V. Улучшение теста

Созданный тест можно улучшить:

- убрать на кадрах поля ввода, в которых мы временно (для проверки работоспособности теста) выводили значение количества правильных ответов;
- сделать отдельный подсчет количества выполненных заданий;
- предусмотреть возможность вывода в кадре результата указания, какие конкретно тестовые задания выполнены правильно, а какие — нет.

33. Выделив первый кадр в таймлайне, изменим скрипт, привязанный к этому первому кадру, на следующий:

```
p_otv=0;
n_otv=0;
k_zadan=0;
pz1=false;
pz2=false;
pz3=false;
stop();
```

```
Начальные присваивания:
– глобальная переменная – количество правильных ответов
– глобальная переменная – количество неправильных ответов
– глобальная переменная – количество заданий
– глобальные переменные логического типа, указывающие правильность
  выполнения каждого задания (по числу имеющихся заданий)

– остановка воспроизведения анимации на первом ее кадре
```

34. На втором кадре удалим поле ввода (экземпляр **rez1**), которое временно использовалось для вывода результатов. Скрипт, присвоенный кнопке проверки, скопируем в буфер и вставим для кнопки “Дальше”, после чего изменим его на следующий:

```
on (press) {
  _root.k_zadan += 1;
  rbprav = _root.rb2.selected;

  if (rbprav) {
    _root.p_otv += 1;
    _root.pz1 = true;
  } else {
    _root.n_otv += 1;
    _root.pz1 = false;
  }
  gotoAndStop(3);
}
```

```
Нажатие кнопки:
– увеличиваем счетчик выполненных заданий
– считываем, выбран ли второй
  переключатель (правильный)
– если true, то
  увеличиваем счетчик правильных ответов,
  отмечаем, что 1-е задание выполнено верно
– иначе
  увеличиваем счетчик неправильных ответов,
  отмечаем, что 1-е задание выполнено
  неверно
– переходим к кадру 3 и останавливаем
  воспроизведение
```

После этого кнопку проверки удалим.

35. На третьем кадре удалим поле ввода (экземпляр **rez2**), которое временно использовалось для вывода результатов. Скрипт, присвоенный кнопке проверки, скопируем в буфер и вставим для кнопки “Дальше”, после чего изменим его на следующий:

```
on (press) {
  _root.k_zadan += 1;
  flag1 = _root.fl1.selected;
  flag2 = _root.fl2.selected;
  flag3 = _root.fl3.selected;
  flag4 = _root.fl4.selected;
  flag5 = _root.fl5.selected;
  prav = !flag1 && flag2 && !flag3 &&
flag4 && flag5;
  if (prav) {
    _root.p_otv += 1;
    _root.pz2 = true;
  } else {
    _root.n_otv += 1;
    _root.pz2 = false;
  }
  gotoAndStop(4);
}
```

```
Нажатие кнопки:
– увеличиваем счетчик выполненных заданий
– считываем по очереди все флажки

– формируем условие правильного ответа
  ("!" – операция "НЕ")
– если true, то
  увеличиваем счетчик правильных ответов,
  отмечаем, что 2-е задание выполнено верно
– иначе
  увеличиваем счетчик неправильных ответов,
  отмечаем, что 2-е задание выполнено
  неверно
– переходим к кадру 4 и останавливаем
  воспроизведение
```

После этого кнопку проверки удалим.

36. На четвертом кадре удалим поле ввода (экземпляр **rez3**), которое временно использовалось для вывода результатов. Скрипт, присвоенный кнопке проверки, скопируем в буфер и вставим для кнопки “Дальше”, после чего изменим его на следующий:

```
on (press) {
  _root.k_zadan += 1;
  totv = _root.txt1.text;

  prav = (totv == "сканер") ||
         (totv == "Сканер") ||
         (totv == "СКАНЕР");
  if (prav) {
    _root.p_otv += 1;
    _root.pz3 = true;
  } else {
    _root.n_otv += 1;
    _root.pz3 = false;
  }
  gotoAndStop(5);
}
```

```
Нажатие кнопки:
– увеличиваем счетчик выполненных заданий
– считываем текст, введенный в поле ввода
  ответа
– перебираем все возможные варианты
  ввода правильного ответа (используется
  логическая операция ИЛИ (|||))
– если true, то
  увеличиваем счетчик правильных ответов,
  отмечаем, что 3-е задание выполнено верно
– иначе
  увеличиваем счетчик неправильных ответов,
  отмечаем, что 3-е задание выполнено
  неверно
– переходим к кадру 5 и останавливаем
  воспроизведение
```

После этого кнопку проверки удалим.

37. На пятом кадре для первого поля ввода (экземпляр **sumkol**) изменим скрипт на следующий:

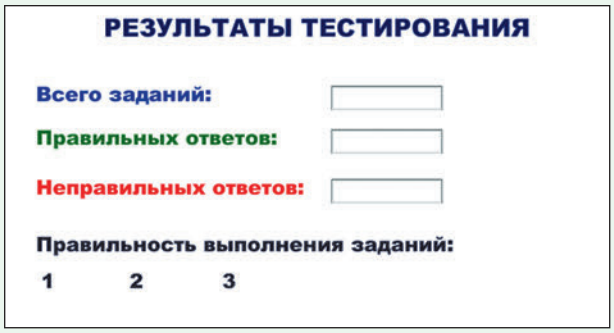
```
onClipEvent(enterFrame) {
  this.text = _root.k_zadan;
}
```

```
Действия при открытии кадра:
– общее число заданий
```

Для кнопки “Повторить тест” изменим скрипт на следующий:

```
on (press) {
  _root.p_otv = 0;
  _root.n_otv = 0;
  _root.k_zadan = 0;
  _root.pz1 = false;
  _root.pz2 = false;
  _root.pz3 = false;
  gotoAndStop(2);
}
```

38. Там же, на пятом кадре, добавим надпись: “Правильность выполнения заданий” и надписи с номерами: “1”, “2”, “3”:



Нарисуем два квадратика — зеленый и красный. Каждый из них преобразуем в библиотечный символ типа MovieClip. На основе этих библиотечных символов создадим на кадре шесть экземпляров (три зеленых, три красных). Дадим им имена экземпляров: **flag1green**, **flag1red**, **flag2green**, **flag2red**, **flag3green**, **flag3red**, соответственно.

После этого (только после именования экземпляров!) расположим их следующим образом:

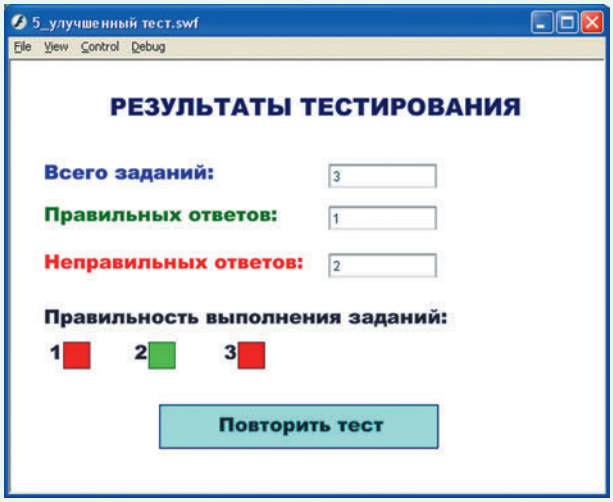
- рядом с номером задания “1” — квадратики **flag1green** и **flag1red** друг поверх друга,
- рядом с номером задания “2” — квадратики **flag2green** и **flag2red** друг поверх друга,
- рядом с номером задания “3” — квадратики **flag3green** и **flag3red** друг поверх друга.

Нарисуем квадратик в любом месте кадра либо вне кадра. Преобразуем его в символ MovieClip. Привяжем к нему следующий скрипт:

```
onClipEvent(load) {
  this._visible = false;
  _root.flag1green._visible = false;
  _root.flag1red._visible = false;
  _root.flag2green._visible = false;
  _root.flag2red._visible = false;
  _root.flag3green._visible = false;
  _root.flag3red._visible = false;
}
onClipEvent(enterFrame) {
  if (_root.pz1) {
    _root.flag1green._visible = true;
  } else {
  _root.flag1red._visible = true;
  }
  if (_root.pz2) {
    _root.flag2green._visible = true;
  } else {
  _root.flag2red._visible = true;
  }
  if (_root.pz3) {
    _root.flag3green._visible = true;
  } else {
  _root.flag3red._visible = true;
  }
}
```

```
Повторяющиеся действия в цикле:
– все добавленные квадратики
  изначально делаем невидимыми

Действия при открытии кадра:
– если первое задание выполнено верно,
  выводим зеленый квадратик
– иначе
  выводим красный квадратик

– если второе задание выполнено верно,
  выводим зеленый квадратик
– иначе
  выводим красный квадратик

– если третье задание выполнено верно,
  выводим зеленый квадратик
– иначе
  выводим красный квадратик
```

39. Запустим анимацию. Проверим ее работу. Теперь при выводе итогового кадра на нем должно отображаться не только общее количество заданий, количество правильных и неправильных ответов, но и индицироваться зелеными и красными квадратиками, какие задания выполнены верно (зеленый квадратик рядом с номером задания), а какие — неверно (красный квадратик).



Улучшить программный код скрипта и сделать его более универсальным по отношению к количеству заданий (без необходимости "размножать" вручную переменные — "флаги" правильности выполнения заданий, соответствующие им операторы **if** в пятом кадре и метки-квадратики) можно, используя массивы и автоматическое создание объектов. В данном занятии эти возможности не рассматриваются.

УРОК 10. ЦИКЛЫ. ОБРАБОТКА ТЕКСТА

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.

Оператор цикла

Циклическими называют действия, выполняемые в программе многократно, в том числе с изменением некоторого параметра (или нескольких параметров) при каждом выполнении этих действий в соответствии с некоторой закономерностью.

Когда флеш-анимация состояла только из одного кадра, мы формировали циклическое (повторяющееся) выполнение действий за счет "зацикливания" самой флеш-анимации: ее единственный кадр многократно вызывается на воспроизведение, если в плеере (по умолчанию) имеется соответствующая настройка. Соответственно, обработчик события **onClipEvent(enterFrame) { *программный код* }** (событие **enterFrame** — "запуск данного кадра на воспроизведение") выполнял функции оператора цикла.

Если же флеш-анимация состоит из нескольких кадров и циклические действия нужно выполнять только на одном из них (т.е. в пределах только одного запуска кадра на воспроизведение), то необходимо использовать операторы цикла.

Цикл с параметром (цикл for)

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, знак неравенства при этом может быть любым). 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--**).

Цикл с предусловием

while (*<условие выполнения цикла>*) { *программный код* }

Программный код в теле цикла выполняется, пока не станет ложным условие в операторе цикла. Причем это условие проверяется *перед* каждым выполнением программного кода в теле цикла, поэтому если условие изначально ложно, такой цикл не будет выполнен ни разу. (При этом в программном коде тела цикла должны происходить какие-то изменения, чтобы в какой-то момент условие завершения цикла стало ложным, иначе цикл будет выполняться бесконечно.)

Пример:

```
i = 0;
while (i < 10) {
    trace(i);    // вывести текущее значение переменной i
    i++;         // изменение значения переменной i
}
```

Цикл с постусловием

do { *программный код* } while (*<условие выполнения цикла>*)

Программный код в теле цикла тоже выполняется, пока не станет ложным условие в операторе цикла. Но в отличие от цикла с предусловием здесь условие проверяется *после* каждого выполнения программного кода в теле цикла, поэтому такой цикл будет выполнен по крайней мере один раз.

Пример:

```
i = 0;
do {
    trace(i);    // вывести текущее значение переменной i
    i++;         // изменение значения переменной i
} while(i < 10);
```

Добавление диалоговых элементов (“компонентов”)

Библиотека элементов диалога раскрывается в отдельном окне выбором в главном меню команды **Window → Components**. В открывшемся окне библиотеки компонентов нужно выбрать раздел **User Interface** (“интерфейс пользователя”):

Основные свойства объектов — диалоговых элементов

Объект	Описание объекта	Свойство	Описание свойства
Button	Кнопка	label	Текст надписи на кнопке
TextInput	Поле ввода текста	text	Текст в поле ввода (может быть задан изначально путем присваивания нужной строки свойству text или получен из поля ввода чтением этого свойства в переменную)
		editable	Возможность изменения содержимого поля (true/false)
		password	Скрывать ли текст в поле заменой символов на “звездочки” или точки, как в полях ввода паролей (true/false)

_visible — видимость объекта (**true** — видимый, **false** — невидимый).

Размерами прямоугольной области диалогового элемента (содержащей надпись рядом с переключателем или флажком) либо размерами поля ввода можно управлять через свойства ширины и высоты, как для любых других объектов Flash.

Обработка текстовых данных

Текстовые (строковые) данные — это наборы символов, записанные в кавычках либо в апострофах (равно допустимы оба варианта).

Текстовая переменная содержит в качестве значения текстовые данные.

Пример:

```
var stroka: String = "Текст";
```

где **String** — это указание, что переменная имеет текстовый тип.

Некоторые специальные символы:

Код	Специальный символ
\n	Новая строка
\\xnn	Символ ASCII с кодом, выраженным шестнадцатеричным числом *nn*
\'	Одиночная кавычка
\"	Двойная кавычка
\\	Одна обратная косая черта

Свойства, операции и методы для работы с текстовыми данными:

конкатенация (соединение) двух строк в одну — операция "сложения" двух строк (оператор **+**): вторая строка дописывается к концу первой (пробелы в месте соединения автоматически не дописываются!). Если одно из "слагаемых" — числовое значение, то оно автоматически преобразуется в текстовую запись;

length — длина строки текста, например: **stroka.length** — длина текстовой строки, хранящейся в переменной **stroka**;

charAt(i) — символ, стоящий в строке текста в позиции с номером **i** от начала (позиции символов в строке нумеруются с нуля), например: **stroka.charAt(i);**

charCodeAt(i) — ASCII код символа, стоящего в строке текста в позиции с номером **i** от начала, например: **stroka.charCodeAt(i);**

toString(i) — попытка преобразовать число — значение переменной **i** в текстовую запись этого числа;

substr(*<начальная позиция>*,*<длина>*) — выделяет из строки подстроку начиная с символа в указанной позиции и с указанной длиной, например, если в переменной **stroka** содержится текст "Привет", запись **stroka.substr(3,2)** вернет строку "ве";

substring(*<начальная позиция>*,*<конечная позиция>*) — выделяет из строки подстроку, начиная с символа в указанной позиции и до символа в указанной конечной позиции (не включая его!), например, если в переменной **stroka** содержится текст "Салют", запись **stroka.substring (2,4)** вернет строку "лю";

> Напомним: позиции символов в строке нумеруются слева направо, начиная с нуля. В методах **substr** и **substring** можно использовать отрицательные номера позиций, тогда нумерация позиций производится справа налево (с конца строки).

indexOf(*<искомая подстрока>*, *<позиция начала поиска>*) — ищет в строке первое вхождение заданной подстроки и возвращает номер позиции в исходной строке, с которой начинается искомая подстрока. Второй параметр (необязательный) — номер позиции (от начала исходной строки), с которого надо начинать поиск;

lastindexOf(*<искомая подстрока>*, *<позиция начала поиска>*) — ищет в строке последнее вхождение (т.е. поиск ведется справа налево) заданной подстроки и возвращает номер позиции в исходной строке, с которой начинается искомая подстрока. Второй параметр (необязательный) — номер позиции (от начала исходной строки), с которого надо начинать поиск, двигаясь к началу строки;

> Если подстрока не найдена в исходной строке, то функции **indexOf** и **lastindexOf** возвращают значение –1.

parseInt(*<строка>*,*<основание системы счисления>*) — процедура, преобразующая строковую запись из символов цифр в соответствующее целое число (в числовой тип), с которым можно производить вычисления. Второй параметр (необязательный, по умолчанию — десятичная система счисления) — система счисления, в которой записано число (основание от 2 до 36);

parseFloat(*<строка>*) — процедура, преобразующая строковую запись из символов цифр в соответствующее вещественное число (в числовой тип), с которым можно производить вычисления;

toLowerCase() — преобразование всех символов строки в строчные, например: **stroka = stroka.toLowerCase();**

toUpperCase() — преобразование всех символов строки в заглавные, например: **stroka = stroka.toUpperCase()**.

Сравнение строк

Строки текста можно сравнивать при помощи операторов сравнения:

!= — несовпадение строк,

== — совпадение строк,

> и **<** — лексикографическое сравнение строк (посимвольно слева направо); меньшей считается строка, в которой символ в очередной сравниваемой позиции имеет меньший код ASCII, либо более короткая строка.

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>**.

Массивы

Массив — набор пронумерованных по порядку переменных (элементов), имеющих одно и то же имя. Обращение к конкретному элементу массива производится по его номеру (индексу).

Создание (инициализация) массива в action script:

1) с заданием исходных значений:

var <имя массива>: Array = [<перечень значений элементов>]; — создается одномерный массив с указанным количеством элементов.

Интересно, что хотя в большинстве языков программирования все элементы массива должны иметь один и тот же тип, в action script это требование не обязательно. Поэтому массивы в action script больше похожи на записи.

var <имя массива>: Array = [[<значения>],[<значения>],...]; — двумерный массив.

Пример: **var matrix: Array = [[1,2,3],[4,5,6]];** — массив из 2×3 (две строки по три элемента в каждой).

2) создание пустого нового массива:

```
var <имя массива>: Array = new Array();
var <имя массива>: Array = [ ];
```

Далее значения элементов задаются при помощи операторов присваивания.

Обращение к элементу одномерного массива:

```
<имя массива>[<индекс>]
```

Обращение к элементу двумерного массива:

```
<имя массива>[<индекс строки>][<индекс элемента в строке>]
```

Нумерация индексов начинается с нуля.

События

onClipEvent(enterFrame) { *программный код* } — обработчик событий, назначаемый символу-клипу или кнопке (символу либо диалоговому элементу Button). Здесь используется событие **enterFrame** — “запуск данного кадра на воспроизведение”.

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.

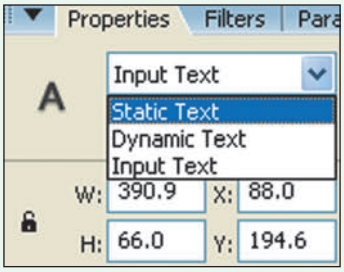
on (press) { *программный код* } — обработчик событий, назначаемый для объекта-мишени. Данное событие также является внешним и выполняется многократно, каждый раз при нажатии левой кнопки мыши на этом объекте.

Динамический текст

Обычная текстовая надпись создается при редактировании кадра Flash и при работе анимации остается неизменной. Такой текст называют “статическим” (**Static Text**).

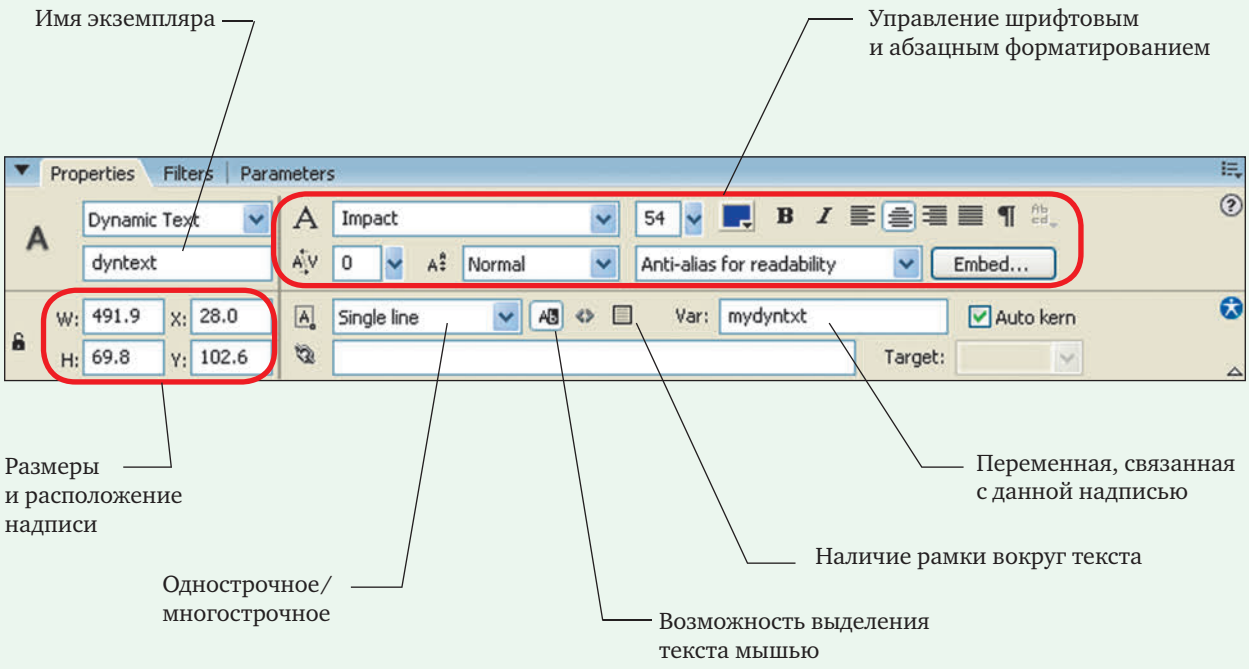
Можно преобразовать надпись в “динамический текст”, в котором собственно текст можно программно менять из скрипта, сохраняя для этого текста изначальное шрифтовое оформление исходной надписи. Для этого нужно в раскрывающемся списке в левой части панели свойств выбрать для надписи режим **Dynamic Text**.

Можно также преобразовать надпись в “пользовательский текст”, в который пользователь при работе анимации может вводить произвольный текст, причем для этого текста сохраняется изначальное шрифтовое оформление исходной надписи. Для этого нужно в раскрывающемся списке в левой части панели свойств выбрать для надписи режим **Input Text**.

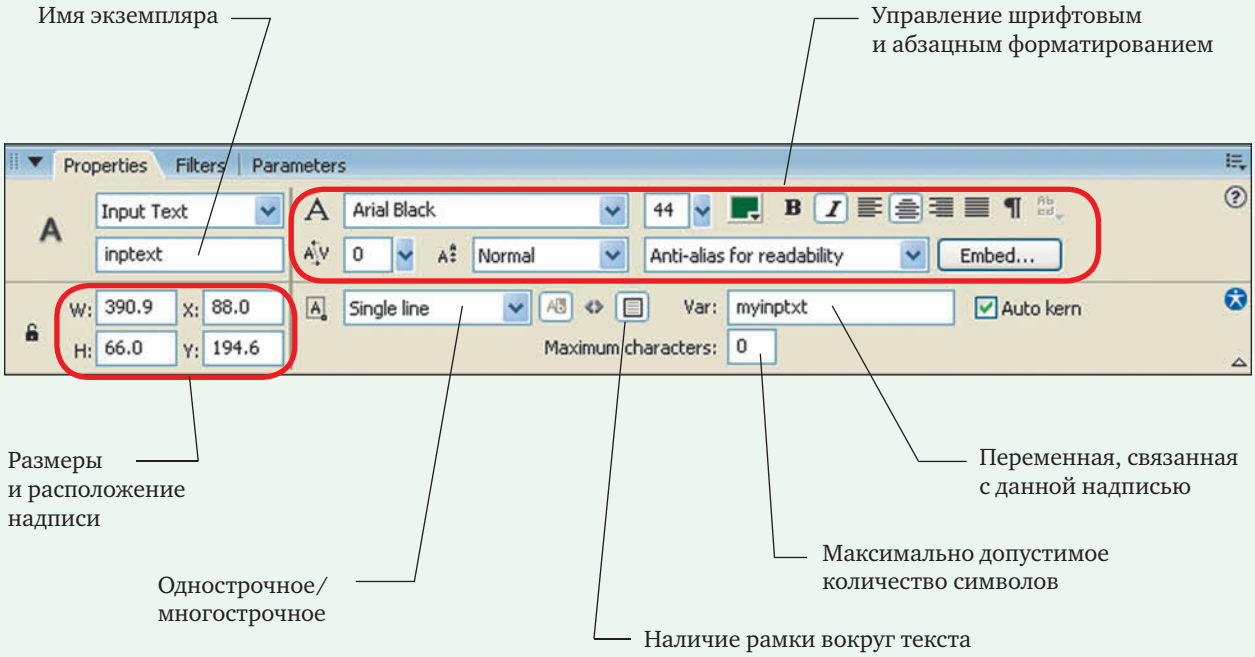


После преобразования надписи в “динамический текст” или в “пользовательский текст” в панели свойств становятся доступны дополнительные элементы настройки:

— для “динамического текста”:



— для “пользовательского текста”:



В списке “Однострочное/многострочное” можно выбрать один из пунктов:

— **Single Line** — однострочная надпись (если текст длиннее, чем отведенная ширина надписи, то часть текста уйдет за границы надписи и будет не видна),

— **Multiline** — многострочная надпись (длинный текст переносится на следующие строки с сохранением ширины надписи, но не ее высоты),

— **Multiline no wrap** — то же, но без автоматического переноса текста,

— **Password** — имеется только для “пользовательского текста”, обеспечивает вывод вводимых с клавиатуры символов в виде “звездочек” (как при вводе пароля).

Кнопка возможности выделения текста мышью — для “динамического текста” позволяет пользователю при работе с анимацией выделять и копировать в буфер текст из надписи (для “пользовательского текста” эта кнопка недоступна, так как данное свойство обеспечивается для таких надписей всегда).

Кнопка наличия рамки вокруг надписи — предписывает выводить вокруг надписи рамку во время выполнения анимации (при редактировании “рабочая” рамка имеется изначально).

Поле максимально допустимого количества символов — для “пользовательского текста” позволяет ограничить длину вводимого текста (чтобы он не выходил за границы однострочной надписи и не переносился на другую строку для многострочной).

Поле **Var** — предназначено для ввода имени переменной, связанной с данной “динамической”, или “пользовательской”, надписью. Эта переменная является глобальной (обращение к ней — через **_root**) и возвращает полную информацию о тексте в данной надписи и о его форматировании (в виде HTML-текста с соответствующими тегами).

К “динамической”, или “пользовательской”, надписи можно обращаться как через эту переменную (например, присваивать ей текстовую строку для отображения этого текста в надписи), так и через свойства экземпляра (тогда обращение производится в виде **<имя_экземпляра>.<свойство>**):

— **.text** — собственно текст в надписи (можно как считывать его в рабочую переменную, так и присваивать этому свойству новое значение в виде текстовой строки; HTML-теги в значении этого свойства отсутствуют);

— **.textColor** — цвет текста. 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

— **.background** — имеет значение **true** или **false**. Значение **false** указывает, что фон надписи — прозрачный. Значение **true** устанавливает непрозрачный белый фон надписи (фон под текстом) и позволяет далее управлять цветом этого фона,

— **.backgroundColor** — цвет фона (действует, если свойство **.background** установлено в **true**, цвет кодируется так же, как и цвет текста — см. выше),

— **.border** — имеет значение **true** или **false**. Значение **false** указывает, что вокруг надписи прямоугольная рамка при работе анимации не выводится. Значение **true** предписывает при работе анимации ограничивать надпись рамкой и позволяет далее управлять цветом этой рамки,

— **.borderColor** — цвет рамки (действует, если свойство **.border** установлено в **true**, цвет кодируется так же, как и цвет текста, — см. выше),

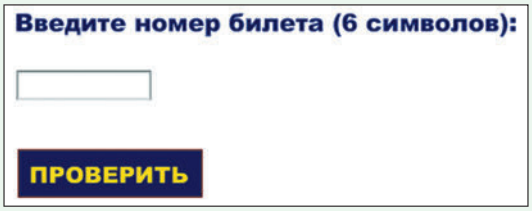
Практические задания

Задание I. Счастливый билет

1. Создадим новую анимацию. Ее первому ключевому кадру присвоим (выделив этот кадр в таймлайне!) скриптовую команду **stop();**.

2. Создадим в этом кадре надпись: “Введите номер билета”. Перетаскивая из панели библиотеки компонентов образец **TextInput**, разместим на кадре поле ввода текста. Назначим ему имя экземпляра **nomer**. Выделив это поле ввода, раскроем в панели параметров (внизу) вкладку **Parameters** и введем значения параметров: **editable = true**, **password = false**, значение параметра **text** оставим пустым.

3. Здесь же нарисуем кнопку “Проверить” и преобразуем ее в библиотечный символ Movie Clip.



4. Создадим два транспаранта (символа MovieClip): “Счастливый билет” и “Несчастливый билет”, присвоим им, соответственно, имена экземпляров **da** и **no**. Каждому присвоим следующий скрипт (один и тот же для обоих транспарантов):

```
onClipEvent(load) {
  this._visible = false;
}
```

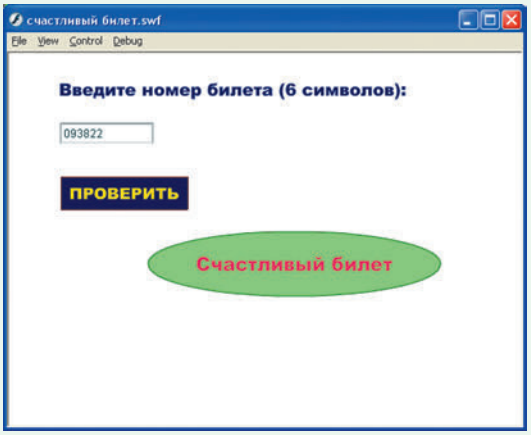
```
Изначально:
– скрываем данный объект;
```

5. Кнопке “Проверить” присвоим следующий программный код:

```
on (press) {
   stroka = _root.nomer.text;

   x1 = parseInt(stroka.substr(0,1)) +
        parseInt(stroka.substr(1,1)) +
        parseInt(stroka.substr(2,1));
   x2 = parseInt(stroka.substr(3,1)) +
        parseInt(stroka.substr(4,1)) +
        parseInt(stroka.substr(5,1));
   if (x1 == x2) {
     _root.da._visible = true;
     _root.no._visible = false;
   } else {
     _root.no._visible = true;
     _root.da._visible = false;
   }
}
```

```
Нажатие кнопки:
– считали в переменную текст
  из поля ввода;
– сумма первых трех цифр номера;


– сумма последних трех цифр номера;


– если суммы первых трех
  и последних трех цифр номера
  равны, то вывести сообщение
  "Счастливый номер" и скрыть
  "Несчастливый номер", иначе –
  наоборот
```

6. Запустим анимацию. Введем номер билета (шесть цифр) и нажмем кнопку “Проверить”. Должна выводиться надпись, указывающая, является ли этот билет счастливым.



Задание II. Палиндром

Палиндром (“перевертыш”) — слово или фраза, которое читается одинаково слева направо и справа налево (во фразе из нескольких слов пробелы при этом не учитываются).

7. Создадим новую анимацию. Ее первому ключевому кадру присвоим (выделив этот кадр в таймлайне!) скриптовую команду `stop();`.

8. В первом ключевом кадре разместим:

— надпись: “Проверка слов на «палиндромность»”;

— надпись: “Введите слово:”;

— поле ввода текста (образец **TextInput**) с именем экземпляра **palin**; ширина — 150, значения параметров:

editable	true
password	false

— кнопку “Проверка” как библиотечный символ Movie Clip;

— две надписи — транспаранта: “Палиндром” и “Не палиндром” (преобразованные в символы Movie Clip) с именами экземпляров **palyes** и **palno** соответственно. Каждому присвоим следующий скрипт (один и тот же для обоих транспарантов):

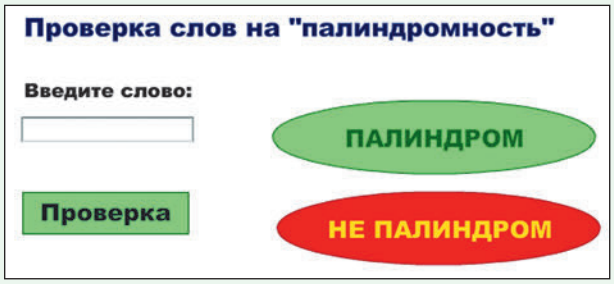
```
onClipEvent(load) {
  this._visible = false;
}
```

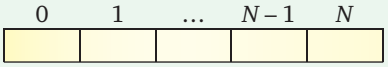
```
Изначально:
– скрываем данный объект;
```

9. Кнопке “Проверить” присвоим следующий программный код:

```
on (press) {
  stroka = _root.palin.text;
  flag = true;
  dl=stroka.length;
  for (i = 0; i <= int(dl/2) – 1; i++) {

    x1 = stroka.substr(i,1);
    x2 = stroka.substr(dl – 1 – i,1);
    if (x1 != x2) {
      flag=false;
    }
  }
  if (flag) {
    _root.palyes._visible = true;
    _root.palno._visible = false;
   } else {
    _root.palno._visible = true;
    _root.palyes._visible = false;
   }
}
```

```
Нажатие кнопки:
– считали текст из поля ввода;
– предполагаем, что палиндром;
– длина строки;
– цикл от первого символа
  до середины строки:
  – первый сравниваемый символ,
  – второй сравниваемый символ,
  – если не равны, то флаг = false,


– конец цикла проверки;
– если флаг остался true,
   то вывести сообщение "Палиндром"
   и скрыть "Не палиндром",
  иначе –
   наоборот
```



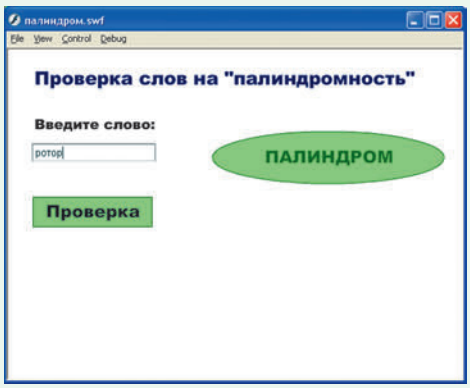


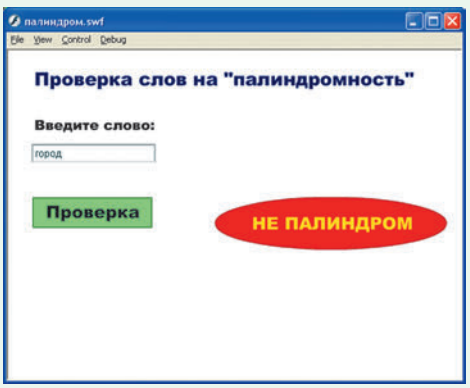
Середина строки — это:
— при четной длине строки — пара символов с номерами $N / 2 – 1$ и $N / 2$;
— при нечетной длине строки — символ с номером $(N – 1) / 2$.

Для проверки, является ли слово палиндромом, нужно по очереди проверять равенство первого символа (с номером 0) и последнего (с номером $N – 1$), второго (с номером 1) и предпоследнего (с номером $N – 2$) и т.д. до середины строки `(int(N/2)–1)`. Очевидно, если определять символ “середины строки” по такой формуле, для четной строки мы получим номер первого символа серединной пары, а для нечетной длины — символ перед средним символом (единственный средний символ проверять на равенство с самим собой нет смысла).

10. Запустим анимацию. Введем слово и нажмем кнопку “Проверить”. Должна выводиться надпись, указывающая, является ли слово палиндромом.

Список слов-палиндромов есть, например, на сайте: *http://www.tramvision.ru/words/pal.htm*. Так, палиндромами являются слова “довод”, “топот”, “ротор” и пр.





Задание III. Азбука Морзе

11. Создадим новую анимацию. Ее первому ключевому кадру присвоим (выделив этот кадр в таймлайне!) скриптовую команду **`stop();`**.

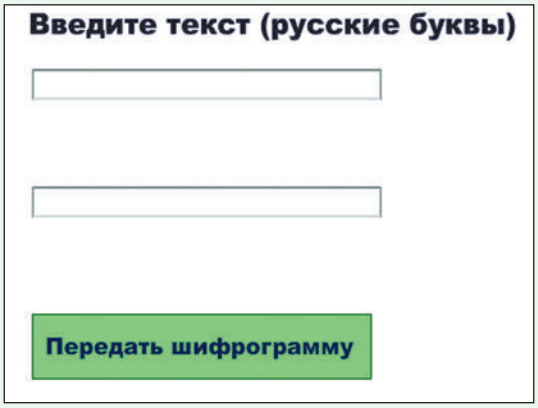
12. В первом ключевом кадре разместим:

— надпись: “Введите текст (русские буквы)”;

— два поля ввода текста (образец **TextInput**) с именами экземпляров **stroka** и **morze** соответственно; ширина — 250, значения параметров:

Имя экземпляра	**stroka**	**morze**
editable	**true**	**false**
password	**false**	**false**

— кнопку “Передать шифрограмму” как библиотечный символ Movie Clip.

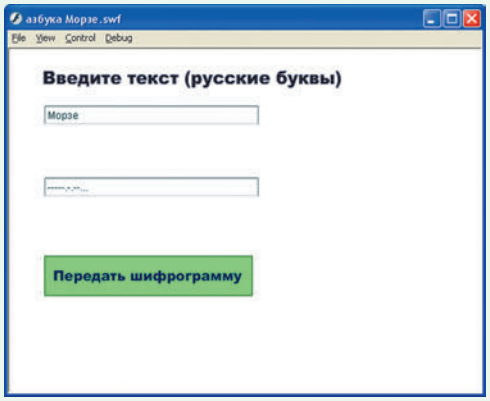


13. Кнопке “Передать шифрограмму” присвоим следующий программный код:

```
on (press) {
var morz:Array = [['А', 'Б', 'В', 'Г', 'Д', 'Е',
'Ж', 'З', 'И', 'К', 'Л', 'М', 'Н', 'О', 'П', 'Р', 'С',
'Т', 'У', 'Ф', 'Х', 'Ц', 'Ч', 'Ш', 'Щ', 'Ъ', 'Ы', 'Ь',
'Э', 'Ю', 'Я'], ['.-', '-...', '.--', '--.', '-..', '.',
'...-', '--..', '..', '-.-', '.-..', '--', '-.', '---',
'.--.', '.-.', '...', '-', '..-', '..-.', '....', '-.-
.', '---.', '----', '--.-', '.--.-.', '-.--', '-..-',
'...-...', '..--', '.-.-']];
  strok = _root.stroka.text;
  strok = strok.toUpperCase();

  mrz = '';
  dl = strok.length;
  for (i = 0; i <= dl - 1; i++) {

    x = strok.substr(i,1);
    for (j = 0; j <= 30; j++) {
      if (x == morz[0][j]) {
        mrz = mrz + morz[1][j];
      }
    }
  }

_root.morze.text = mrz;
}
```

```
Нажатие кнопки:
- массив из двух строк:
  в первой строке перечислены
  русские заглавные буквы,
  во второй строке — коды Морзе
  для каждой из этих букв.
  Букве с индексом [1,i]
  соответствует код Морзе
  с индексом [2,i];
- считали текст из поля ввода;
- преобразовать буквы в
заглавные;
- пустая строка для кода Морзе;
- длина строки;
- цикл от первого символа
  до конца строки:
  - очередной символ строки,
  - цикл перебора букв в массиве,
  - если очередной символ = букве,
    то ее код дописать к строке,

- конец цикла перебора букв,
- конец цикла просмотра строки;

- вывести накопленную строку
  кода Морзе
```

14. Запустим анимацию. Введем в первое поле слово и нажмем кнопку “Передать шифрограмму”. Во втором поле появится соответствующий код Морзе. (Латинские буквы, цифры, пробелы и прочие знаки в исходной строке пропускаются.)



Задание IV. Поиск ключевых слов

Эта операция позволяет обрабатывать ответы на тесты с вводом ответа в свободной форме — отыскивать требуемый “ключевой” набор символов во введенной пользователем строке. Например, на вопрос “Какое устройство предназначено для ввода в компьютер графических изображений с листов бумаги?” пользователь может ввести в ответ фразу “Это устройство — сканер.”, из которой можно извлечь “ключевое” слово “сканер”.

15. Создадим новую анимацию. Ее первому ключевому кадру присвоим (выделив этот кадр в таймлайне!) скриптовую команду **stop();** .

16. В первом ключевом кадре разместим:

— вопрос теста, например: “Какое устройство предназначено для ввода в компьютер графических изображений с листов бумаги?”;

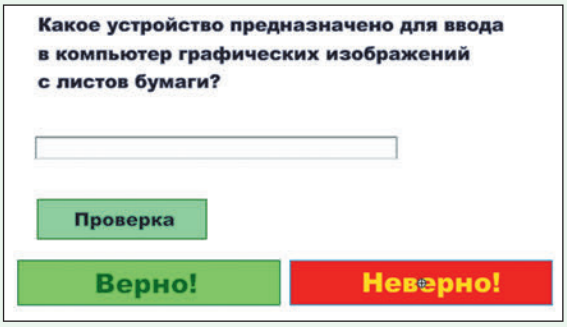
— поле ввода текста (образец **TextInput**) с именем экземпляра **otvet**; ширина — 350, значения параметров: **editable** = **true**, **password** = **false**;

— кнопку “Проверка” как библиотечный символ Movie Clip;

— две надписи — транспаранта: “Верно!” и “Неверно!” (преобразованные в символы Movie Clip) с именами экземпляров **otyes** и **otno** соответственно. Каждому присвоим следующий скрипт (один и тот же для обоих транспарантов):

```
onClipEvent(load) {
  this._visible=false;
}
```

```
Изначально:
— скрываем данный объект;
```

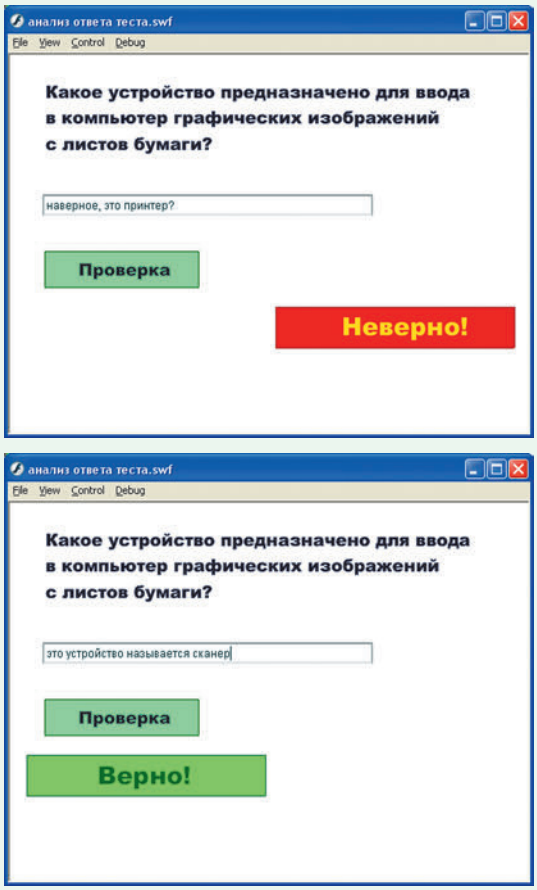


17. Кнопке “Проверить” присвоим следующий программный код:

```
on (press) {
  stroka = _root.otvet.text;
  stroka = stroka.toUpperCase();
  votv1 = 'СКАНЕР';
  votv2 = 'СКАННЕР';
  podstroka1 = stroka.indexOf(votv1);
  podstroka2 = stroka.indexOf(votv2);
  if ((podstroka1 != -1) ||
      (podstroka2 != -1)) {
    _root.otyes._visible = true;
    _root.otno._visible = false;
   } else {
    _root.otno._visible = true;
    _root.otyes._visible = false;
   }
}
```

```
Нажатие кнопки:
— считали текст из поля ввода;
— преобразовать буквы в заглавные;
— первый возможный вариант ответа;
— второй возможный вариант ответа;
— ищем в строке первый ответ;
— ищем в строке второй ответ;
— если хотя бы один вариант ответа
  найден (возвращено не "-1"),
   то вывести сообщение "Верно"
   и скрыть "Неверно",
  иначе —
   наоборот
```

18. Запустим анимацию. Введем свой ответ и нажмем кнопку “Проверка”. Должен выдаваться соответствующий результат:



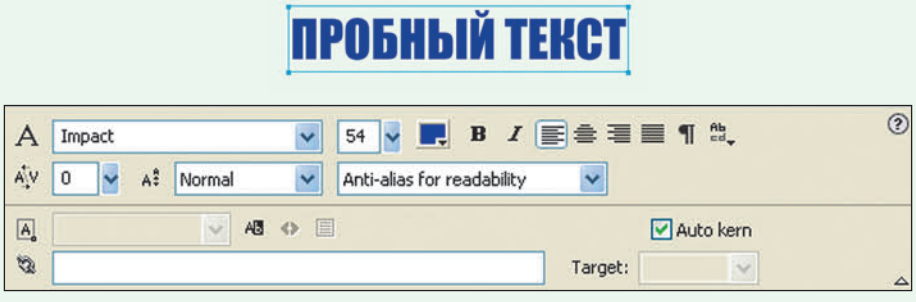
В данном случае предусмотрен поиск во введенной строке двух возможных вариантов ответа: “сканер” и “сканнер” (с учетом возможной ошибки написания). Возможен и поиск части слова (например, без учета окончания), чтобы проверять ответ независимо от склонения/спряжения/падежа.

Эти “ключевые” слова и ищутся во введенной строке. Метод **indexOf** возвращает либо номер позиции в строке, начиная с которой в ней находится искомая подстрока, либо значение –1, если подстрока не найдена. Номер позиции найденной подстроки нам не важен, мы используем лишь значение –1 как признак отсутствия в исходной строке того или иного “ключевого” слова. Если хотя бы для одного варианта ответа результат его поиска в исходной строке НЕ равен –1, то это признак того, что введенная строка содержит в себе правильный ответ. (Условия сравнения объединены логической операцией ИЛИ: в строке достаточно обнаружить хотя бы один вариант правильного “ключевого” слова.)

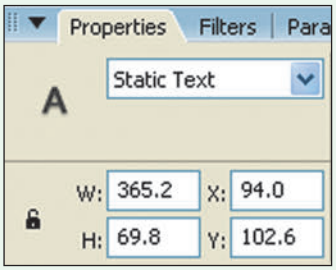
Задание V. Работа с “динамическим текстом” и “пользовательским текстом”

Недостатком текстового объекта **`TextInput`** является невозможность изменения параметров шрифта, которым выводится текст. Альтернативным вариантом является использование собственных текстовых объектов Flash, аналогичных обычным текстовым надписям, но с возможностью менять их текст. Такие объекты называются “динамическим текстом” (**Dynamic Text**).

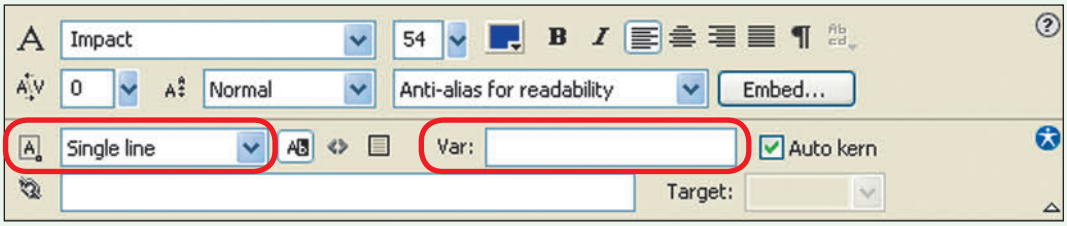
19. Создадим новую анимацию. Создадим на ней текстовую надпись с произвольным текстом (например, “ПРОБНЫЙ ТЕКСТ”) и зададим для нее желаемое шрифтовое оформление.



20. Обратим внимание на левую часть панели свойств созданного текстового объекта (когда он выделен):



В раскрывающемся списке по умолчанию выбран пункт **Static Text** (“статичный текст”). Выберем вместо него пункт **Dynamic Text** (“динамический текст”). Сама надпись не изменилась, но в панели свойств объекта появились новые элементы:

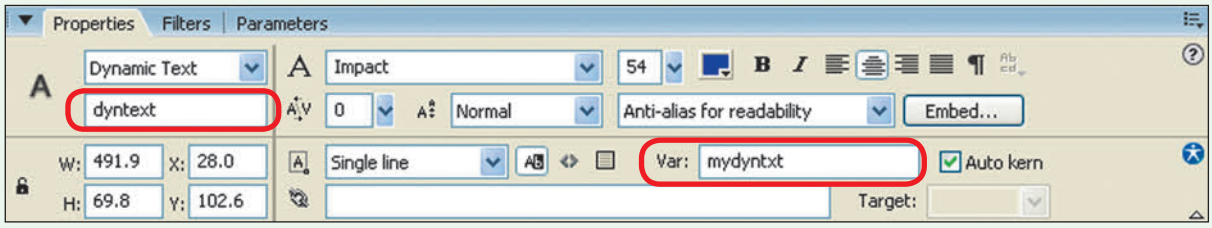


Нам важны из них два: раскрывающийся список, расположенный справа от значка , и поле **Var**.

В раскрывающемся списке выбирается тип “динамического текста”: однострочный (**Single Line**) или многострочный (**Multiline** либо **Multiline no wrap**). Если длина текста, который будет выводиться в этом объекте, гарантированно не будет превышать длину изначально введенного в надписи текста, то можно оставить этот текст однострочным. Однако если позже в него потребуется вывести более длинный текст, часть его, выходящая за пределы изначального размера надписи, окажется не видна. Если же выбрать многострочный вариант, то лишний текст окажется перенесен на следующую строку (как, например, в надписях в MS Office), но тогда новая строка текста может наползать на расположенное ниже изображение. Наилучшим может быть выбор однострочного варианта с изначальным выбором достаточной ширины текста (за счет вбивания пробелов в изначальную надпись или изменения ширины рамки вокруг текста при его выделении перетаскиванием мышью правой и/или левой границы рамки за квадратный маркер в ее середине).

В поле **Var** вводится имя переменной, которая будет связана с данным “динамическим текстом” в скриптовой программе. Считывая значение этой переменной, можно получить текстовую строку, которая выведена на кадр в качестве “динамического текста”. Присвоив же этой переменной другую текстовую строку, можно вывести соответствующий текст, который “унаследует” все шрифтовые настройки, сделанные для изначальной надписи.

21. Введем для созданной “динамической надписи” имя экземпляра **dyntext**, а в качестве имени переменной введем **mydyntxt**.



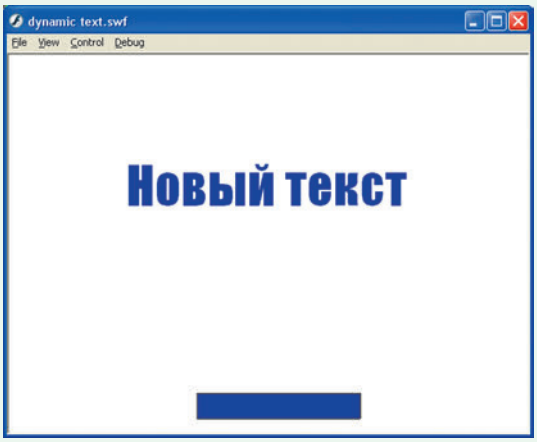
22. Нарисуем прямоугольную кнопку, преобразуем ее в объект MovieClip и привяжем к этой кнопке скрипт:

```
on (press) {
  _root.mydyntxt = 'Новый текст';
}
```

```
Нажатие кнопки:
— присваиваем переменной текстовую строку
```

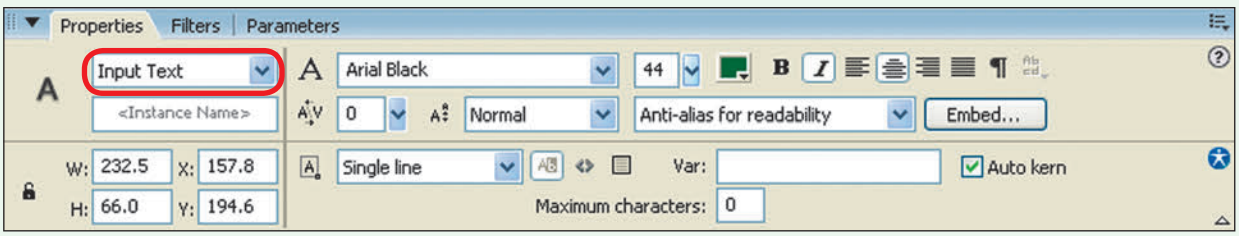
Переменная, присвоенная “динамическому тексту”, является глобальной.

23. Запустим анимацию. При нажатии мышью на заготовленную кнопку текст в надписи изменится на заданный нами. При этом все заданное нами шрифтовое оформление сохранится.

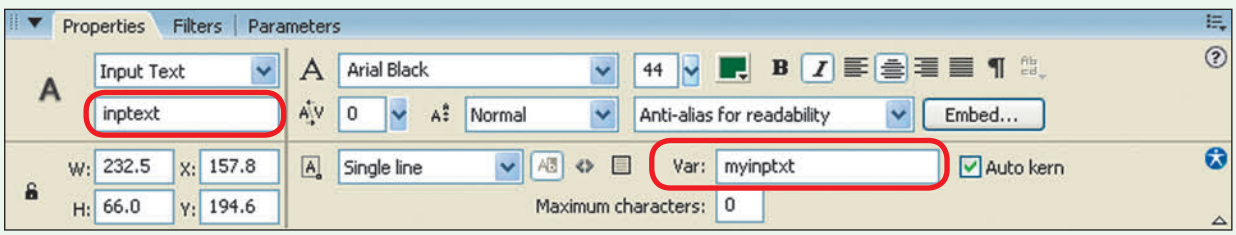


24. Добавим под уже имеющейся надписью еще одну надпись с текстом “Фамилия” и зададим для нее желаемое шрифтовое оформление:

25. Для созданной надписи выберем вместо **Static Text** пункт списка **Input Text** (“текст ввода”, или “пользовательский текст”). Панель свойств примет вид:



Так же, как мы делали для “динамического текста”, увеличим ширину поля этой надписи, зададим для нее имя экземпляра **inptext**, а имя переменной — **myinptxt**. Дополнительно нажмем кнопку для отображения прямоугольной рамки вокруг надписи, чтобы видеть ее размеры.

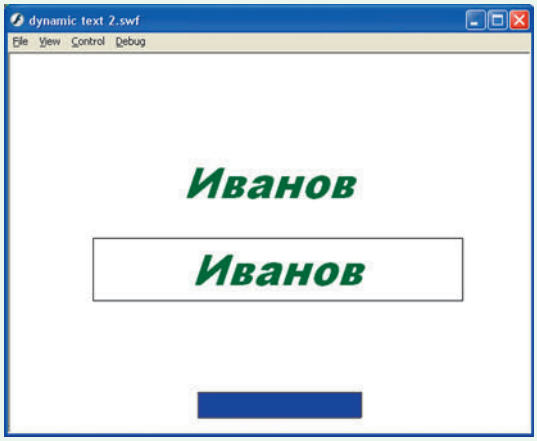


26. Изменим текст скрипта, присвоенный кнопке:

```
on (press) {
  tt = _root.myinptxt;

  _root.mydyntxt = tt;
}
```

```
Нажатие кнопки:
— считываем в переменную tt текст
  из "пользовательского текста"
— присваиваем переменной текстовую строку
```

27. Попробуем запустить анимацию. Щелкнув мышью на надписи “Фамилия”, отредактируем этот текст — сотрем его и введем свою фамилию. При нажатии кнопки результат окажется следующим: вместо прежней “динамической надписи” появится текст, введенный в “пользовательскую надпись”, но вместе с его шрифтовым оформлением!



Причина — в том, что переменная, сопоставленная надписи, содержит в себе не просто текст, а целую HTML-строку с тегами, кодирующими оформление этого текста!

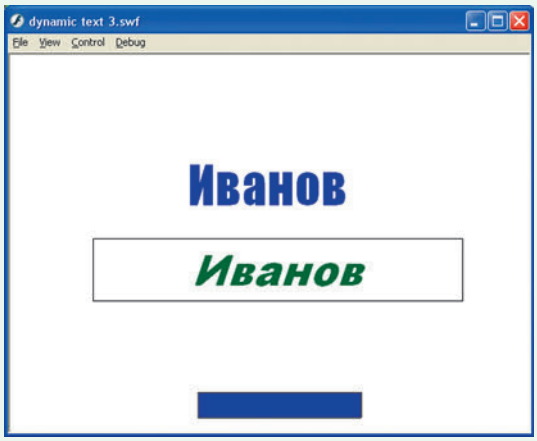
```
<p align="center"><font face="Arial Black" size="44" color="#006633"
letterSpacing="0.000000" kerning="1">
<i>Фамилия</i></font></p>
```

28. Изменим текст скрипта — будем обращаться к “пользовательской надписи” через имя экземпляра и свойство **text**:

```
on (press) {
  tt = _root.inptext.text;

  _root.mydyntxt = tt;
}
```

```
Нажатие кнопки:
– считываем в переменную tt текст
  из "пользовательского текста"
– присваиваем переменной текстовую строку
```

29. Запустим анимацию. Введем в “пользовательский текст” свою фамилию и нажмем кнопку. Теперь текст, введенный в “пользовательскую надпись”, отображается в “динамической надписи” с сохранением прежнего шрифтового оформления последней.

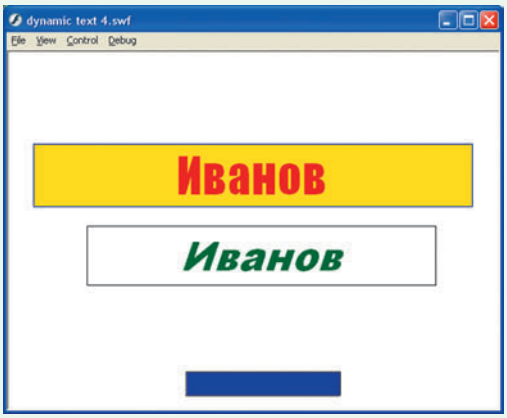


30. Аналогичным способом можно управлять другими свойствами “динамической надписи”, или “пользовательской надписи”. Например:

```
on (press) {
  tt = _root.inptext.text;

  _root.dyntext.text = tt;
  _root.dyntext.textColor = 0xFF0000;
  _root.dyntext.background = true;
  _root.dyntext.backgroundColor = 0xFFFF00;
  _root.dyntext.border = true;
  _root.dyntext.borderColor = 0x0000FF;

}
```

```
Нажатие кнопки:
– считываем в переменную tt текст
  из "пользовательского текста"
– присваиваем надписи текст
– устанавливаем красный цвет текста
– включаем цветной фон текста
– устанавливаем желтый цвет
  фона текста
– включаем рамку вокруг текста
– включаем синий цвет рамки
```

31. Запустим анимацию. Введем в “пользовательский текст” свою фамилию и нажмем кнопку. Получим следующий результат:



УРОК 11. РИСОВАНИЕ: ГРАФИЧЕСКИЕ ПРИМИТИВЫ

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(enterFrame) { *программный код* } — обработчик событий, назначаемый символу-клипу или кнопке (символу либо диалоговому элементу Button). Здесь используется событие **enterFrame** — “запуск данного кадра на воспроизведение”.

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.

on (press) { *программный код* } — обработчик событий, назначаемый для объекта-мишени. Данное событие также является внешним и выполняется многократно, каждый раз при нажатии левой кнопки мыши на этом объекте.

this.onMouseDown = function () { *программный код* } — обработчик события “нажатие кнопки мыши”. Размещается внутри обработчика **onClipEvent(enterFrame) { }** и срабатывает в момент нажатия левой кнопки мыши (именно при ее нажатии, удерживание кнопки нажатой не вызывает данное событие).

this.onMouseMove = function() { *программный код* } — обработчик события “перемещение мыши”. Размещается внутри обработчика **onClipEvent(enterFrame) { }**.

this.onMouseUp = function(){ *программный код* } — обработчик события “отпускание кнопки мыши”. Размещается внутри обработчика **onClipEvent(enterFrame) { }**.

Свойства объекта (клипа, кнопки)

_x, _y — координаты объекта (в пикселях);
_root._xmouse, _root._ymouse — ширина и высота объекта (в пикселях);
Mouse.hide() — спрятать “стандартный” для Windows курсор мыши.

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);

.beginFill(<*цвет*>,<*прозрачность*>) — устанавливает для рисуемой фигуры фоновое закрашивание. Параметры:

— **<*цвет*>** — кодируется так же, как цвет рисования линий, значение **undefined** — отсутствие фонового закрашивания;

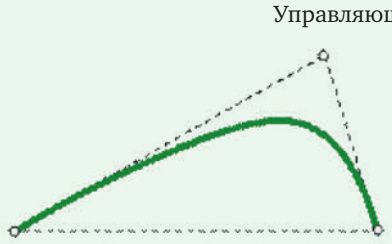
— **<*прозрачность*>** — значение от 0 (полная прозрачность) до 100 (полная непрозрачность);

Для создания замкнутой фигуры достаточно при рисовании отдельными отрезками соединить последнюю точку с первой (т.е. закончить рисование в конечной точке).

.endFill() — прекратить дальнейшее использование фоновой заливки. Если последняя рисуемая фигура при этом была не замкнута, то она автоматически преобразуется в замкнутую (последняя точка соединяется отрезком с первой) и закрашивается с последними параметрами фоновой заливки, после чего действие заливки прекращается;

Чтобы не было путаницы, рекомендуется при рисовании замкнутых фигур с закрашиванием фона ВСЕГДА сначала включать фоновое закрашивание командой **.beginFill(…)**, а после завершения рисования фигуры ВСЕГДА выключать фоновое закрашивание командой **.endFill()**.

.curveTo(x,y,x0,y0) — кривая линия Безье: *x*, *y* — координаты конечной точки рисования, *x*0, *y*0 — координаты "управляющей точки", определяющей кривизну:



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);

`.graphics.beginFill``(...)` — задание цвета и прозрачности фоновой закраски (см. выше);

`.graphics.endFill()` — завершение фоновой закраски (см. выше);

`.graphics.curveTo(x,y,x0,y0)` — кривая Безье (см. выше);

`.graphics.clear()` — очистка “холста” (см. выше);

`.graphics.drawRect(x,y,w,h)` — рисование прямоугольника: *x*, *y* — координаты начальной точки (верхний левый угол), *w* — ширина, *h* — высота;

`.graphics.drawRoundRect(x,y,w,h,sx,sy)` — рисование прямоугольника со скругленными углами: *x*, *y*, *w*, *h* — аналогично обычному прямоугольнику (выше), *sx* и *sy* — значения скругления углов по осям x и y соответственно;

`.graphics.drawCircle(x,y,R)` — рисование окружности: *x*, *y* — координаты центра, *R* — радиус;

`.graphics.drawEllipse(x,y,w,h)` — рисование эллипса. Указываются координаты верхнего левого угла, ширина и высота воображаемого прямоугольника, в который вписан эллипс.

Реализовать рисование в ActionScript 2.0 фигур-примитивов, реализованных в ActionScript 3.0, можно, создав свой собственный класс graphics. О том, как это сделать (включая требуемый программный код, реализующий этот класс), см. на сайте *http://edapskov.ru/pages.php?id=58*.

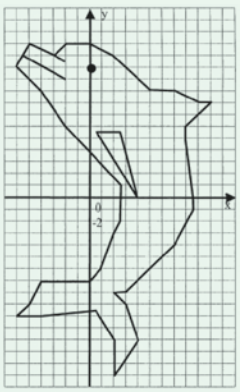
Практические задания

Задание I. Рисование по координатам

Подобные задания часто используют при изучении графических возможностей различных языков программирования. Сначала на клетчатой бумаге рисуется координатная плоскость и вручную изображается желаемая картинка путем соединения отрезками точек на пересечении клетчатой сетки. Затем выполняется кодирование полученного рисунка, при этом определяется точка начала рисования и далее программируется вычерчивание контура “без отрыва пера от бумаги”. При необходимости затем ставится следующая начальная точка, вычерчивается следующий контур и т.д.

При этом нужно помнить, что в компьютерной системе координат ось *Y* направлена вниз, поэтому если на исходном рисунке ось *Y* направлена вверх, нужно менять знак всех координат *y* на противоположный.

Создадим рисунок дельфина, используя исходный рисунок:



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

2. Нарисуем в первом кадре прямоугольник с выключенным фоном. Преобразуем его в символ MovieClip (это будет наш “холст”). При этом точку привязки MovieClip разместим в середине, так как в исходном рисунке начало системы координат располагается примерно посередине. Размеры MovieClip на кадре зададим так, чтобы влево, вправо, вверх и вниз от середины объекта было достаточное пространство (определяется максимальным абсолютным значением координаты *x* и координаты *y* точек рисунка, соответственно). В данном случае установим размер MovieClip равным 400 × 400 (при необходимости надо увеличить размеры кадра анимации).

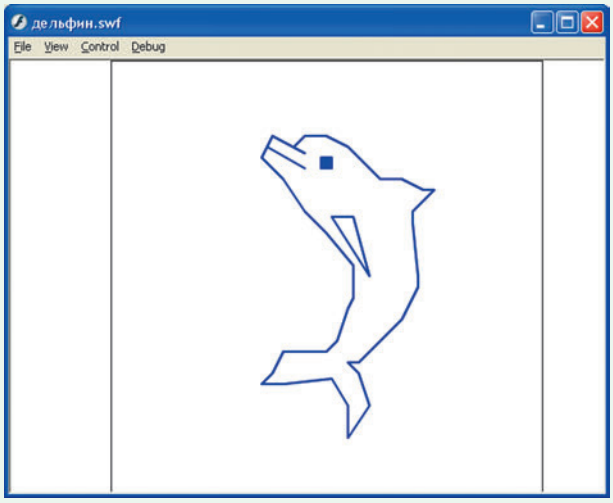
3. Кодируем рисунок по точкам, меняя знак координат *y* и увеличивая их значения в 10 раз. Получим следующую запись:

```
this.lineStyle(2,0x0000FF, 100);                  — синяя линия толщиной 2 пикселя

this.moveTo(-20,-100);                            — часть "клюва"
this.lineTo(-55,-120);
this.lineTo(-50,-130);
this.lineTo(-20,-114);

this.moveTo(-30,-120);                            — туловище
this.lineTo(-20,-130);
this.lineTo(0,-130);
this.lineTo(20,-120);
this.lineTo(50,-90);
this.lineTo(70,-90);
this.lineTo(90,-80);
this.lineTo(100,-80);
this.lineTo(80,-60);
this.lineTo(80,-50);
this.lineTo(85,0);
this.lineTo(85,10);
this.lineTo(70,40);
this.lineTo(30,80);
this.lineTo(20,80);
this.lineTo(30,90);
this.lineTo(40,120);
this.lineTo(20,150);
this.lineTo(20,120);
this.lineTo(5,95);
this.lineTo(-40,100);
this.lineTo(-60,100);
this.lineTo(-50,90);
this.lineTo(-40,70);
this.lineTo(0,70);
this.lineTo(10,60);
this.lineTo(20,30);
this.lineTo(25,20);
this.lineTo(25,-10);
this.lineTo(0,-40);
this.lineTo(-20,-60);
this.lineTo(-40,-90);
this.lineTo(-60,-110);
this.lineTo(-55,-120);

this.moveTo(5,-55);                               — плавник
this.lineTo(25,-55);
this.lineTo(40,0);
this.lineTo(5,-55);

this.moveTo(-5,-100);                             — глаз
this.beginFill(0x0000FF,80);
this.lineTo(5,-100);
this.lineTo(5,-110);
this.lineTo(-5,-110);
this.lineTo(-5,-100);
this.endFill();
```

3. Присвоим этот программный код объекту MovieClip (“холсту”), поместив его внутрь вызова обработчика события `load`:

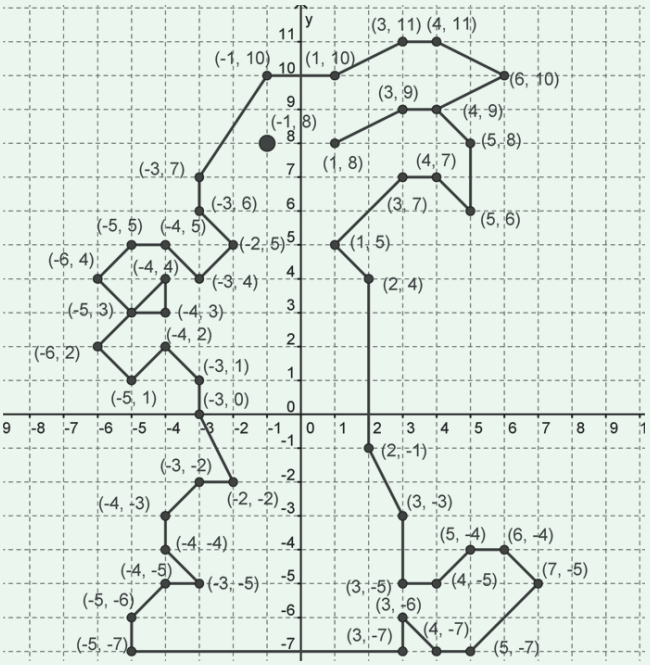
```
onClipEvent(load) {
...
}
```

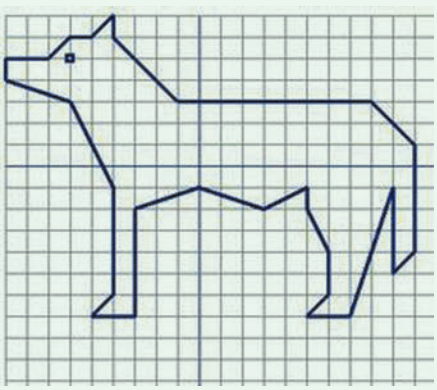
4. Запустим анимацию. Должен быть выведен рисунок:

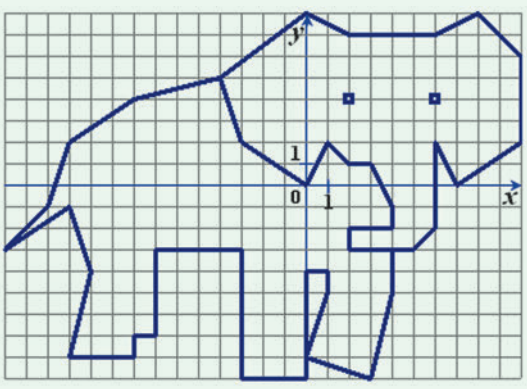


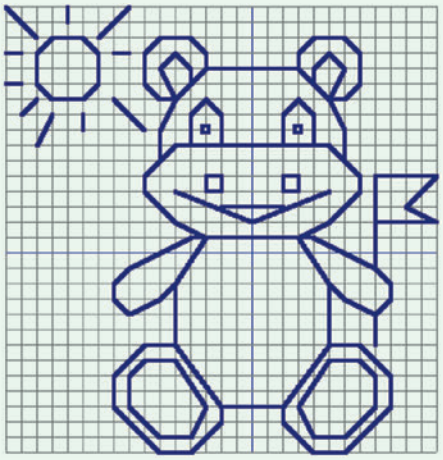
При необходимости проверяем и корректируем код.

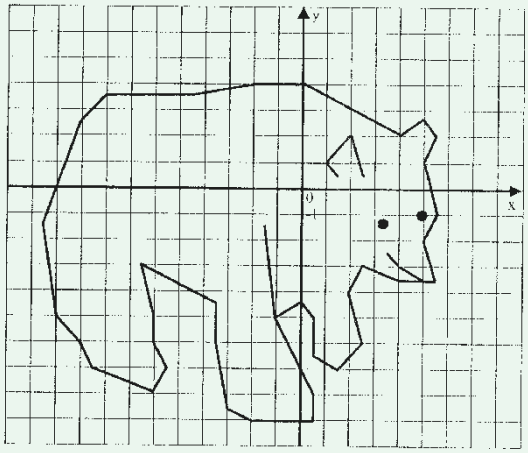
5. Самостоятельно создайте рисунки по следующим заготовкам либо по схемам рисования, выполненным самостоятельно:

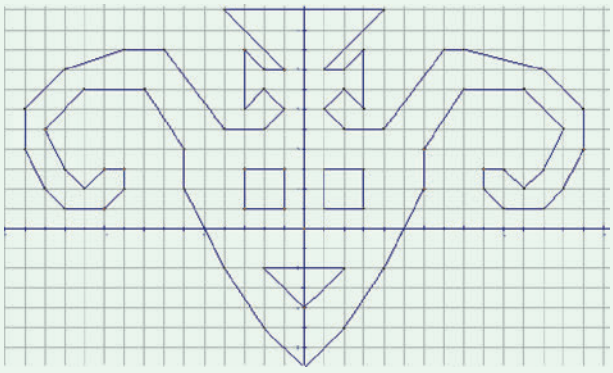












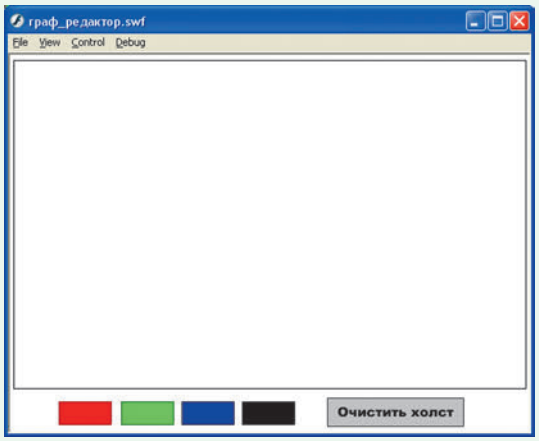
Задание II. Графический редактор

Цель задания — создать простейший “графический редактор”, позволяющий рисовать инструментом “Карандаш” с возможностью выбора цвета из представленной палитры.

6. Создадим новую анимацию. Нарисуем в ее первом кадре прямоугольник размером во весь кадр с выключенным фоном. Преобразуем его в символ MovieClip, точку привязки разместим в верхнем левом углу (это будет наш “холст”). Масштабируем “холст” так, чтобы получить небольшой отступ внизу в виде небольшой горизонтальной полоски для размещения палитры цветов. Назначим ему имя экземпляра: **holst**.

ВАЖНО! Необходимо сначала нарисовать прямоугольник размером во весь кадр, и только после этого можно, преобразовав его в MovieClip, выполнить масштабирование этого объекта, сделав отступ его нижней границы от нижнего края кадра.

7. Под “холстом” нарисуем несколько прямоугольников с разными цветами фоновой закраски: красный, зеленый, синий, черный, а также кнопку “Очистить холст”. Преобразуем каждый из них в объект MovieClip.



8. Назначим для первого кадра программный код, определяющий глобальные переменные: цвет рисования и текущее состояние “пера” (“поднято”/“опущено”, т.е. перемещение курсора мыши с “Карандашом” без рисования или с рисованием соответственно):

```
clr = 0x000000;          — изначально цвет рисования — черный
pero = false;            — изначально "перо" поднято
linia = 2;               — толщина линии — два пикселя
h0 = 400;                — высота кадра анимации
```

9. Назначим для каждой из созданных кнопок — образцов цвета соответствующий программный код:

— красная кнопка:

```
on (press) {                          — при нажатии включить красный цвет рисования
  _root.clr = 0xFF0000;
}
```

— зеленая кнопка:

```
on (press) {                          — при нажатии включить зеленый цвет рисования
  _root.clr = 0x00FF00;
}
```

— синяя кнопка:

```
on (press) {                          — при нажатии включить синий цвет рисования
  _root.clr = 0x0000FF;
}
```

— черная кнопка:

```
on (press) {                          — при нажатии включить черный цвет рисования
  _root.clr = 0x000000;
}
```

10. Назначим для кнопки "Очистить холст" программный код:

```
on (press) {                          — при нажатии вызвать команду стирания рисунка
  _root.holst.clear();
}
```

11. Загрузим в Flash изображение карандаша ( , см. урок 7). Преобразуем его в объект MovieClip (точка привязки — в верхнем левом углу!). Дадим ему имя экземпляра **cursr**.

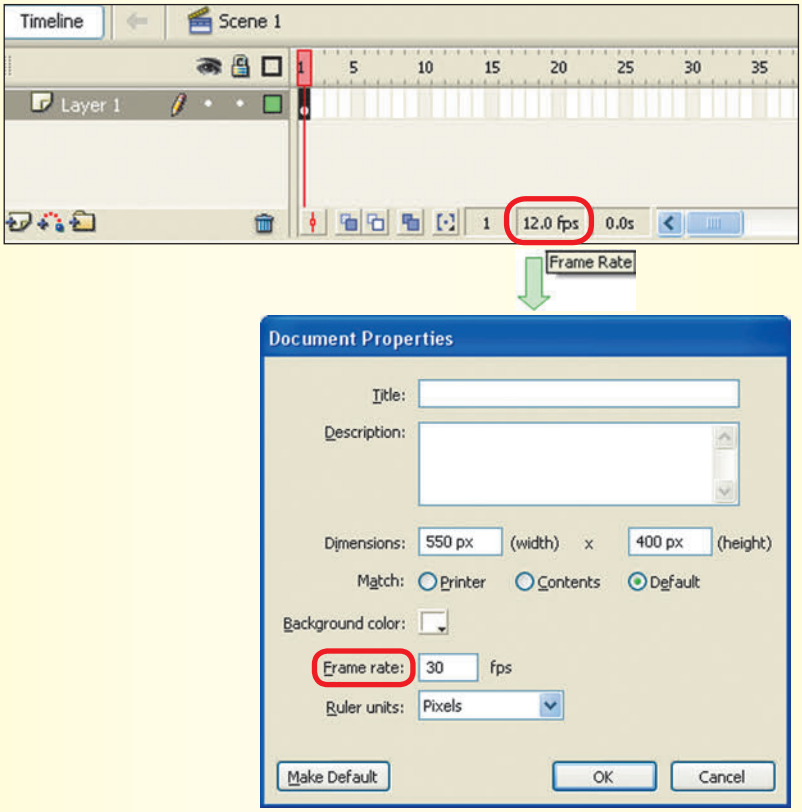
Привяжем к этому объекту (новому курсору мыши) программный код:

```
onClipEvent(load) {                   Изначально:
  Mouse.hide();                       — скрыть настоящий курсор мыши
}
onClipEvent(enterFrame) {             Повторение действий:
this.onMouseDown = function () {      Обработчик нажатия кнопки мыши:
 _root.pero = true;                   — "перо опущено" — рисуем,
 _root.holst.lineStyle(_root.linia,_root.clr);  — выбор толщины и цвета линии,
 y = _root._ymouse/_root.holst._height*_root.h0;  — пересчет координаты y,
 _root.holst.moveTo(_root._xmouse,y);  — текущая точка — в месте нажатия
}                                     кнопки мыши

this.onMouseMove = function() {       Обработчик движения мыши:
 if (_root.pero) {                    — если перо опущено, то рисуем:
 y = _root._ymouse/_root.holst._height*_root.h0;  — пересчет координаты y,
 _root.holst.lineTo(_root._xmouse,y);  — линия до текущего места курсора,
 }
}
this.onMouseUp = function() {         Обработчик отпускания мыши:
 _root.pero = false;                  — "перо поднято" — не рисуем
}
 this._x = _root._xmouse;             Перемещение курсора (карандаша)
 this._y = _root._ymouse;             вслед за мышью
}
```

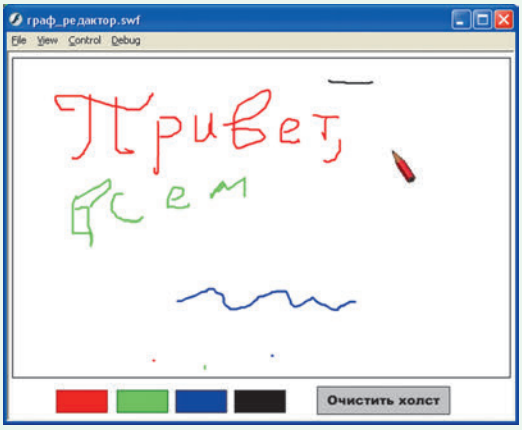
Смысл пересчета координаты **y**: мы определяем координату курсора мыши **_ymouse** в системе координат, связанной со всем кадром анимации, а рисование производится в системе координат, связанной с объектом-"холстом". Но последний имеет меньшую высоту, и это различие надо скомпенсировать исходя из пропорции — отношения высоты кадра к высоте "холста".

Показанный пересчет координат рассчитан на описанный выше случай, когда холст вплотную прилегает к верхнему, левому и правому краям, а отступ сделан только снизу. Если имеется отступ холста от верхнего края кадра, то нужно при пересчете координаты **y** вычесть значение этого отступа. Если есть отступы краев холста от левого и правого краев кадра, то нужно аналогичным способом выполнять компенсационный пересчет и координаты **x**.

Чтобы при движении мыши объект-курсор двигался более плавно, надо увеличить частоту кадров, вызвав диалоговое окно двойным щелчком мыши на обозначении частоты кадров под таймлайном.



12. Запустим анимацию. Проверим ее работу. Когда левая кнопка мыши не нажата, курсор в виде карандаша просто движется по “холсту”. При нажатии левой кнопки мыши и ее движении должно начаться рисование линии из отрезков прямых, пока левая кнопка мыши удерживается нажатой. При отпускании левой кнопки мыши рисование прекращается и снова начинается при повторном нажатии кнопки мыши. После щелчка мышью на одном из образцов цвета последующее рисование выполняется уже этим цветом (изначально — черный). При щелчке на кнопке “Очистить холст” все нарисованное стирается.



Самостоятельно попытайтесь реализовать кнопки выбора инструмента рисования: “Карандаш” / “Кисть”. При этом курсор надо оставить в виде “карандаша”, но по нажатию на соответствующей кнопке менять толщину линии (“карандаш” — два пикселя, “кисть” — пять пикселей).

Окончание читайте в следующем номере.

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** День классного руководителя 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 **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

IX Межрегиональная олимпиада школьников по информатике и компьютерной безопасности

П.Ю. Селиванов,
Москва

IX Межрегиональная олимпиада школьников по информатике и компьютерной безопасности проводилась в два тура. Первый тур проходил с 15 сентября по 19 октября 2014 года в дистанционной форме на интернет-сайте *www.v-olymp.ru*.

Второй тур для 11-х классов проводился в очной форме 26 октября 2014 года в Москве и Санкт-Петербурге. Школьники 9–10-х классов получили возможность поучаствовать в мероприятии в этот же день, но дистанционно на интернет-сайте *www.v-olymp.ru*.

Всего в олимпиаде приняли участие более 400 человек. По результатам проверки работ заключительного тура и с учетом апелляции 18 человек были награждены дипломами I, II, III степени и ценными призами. Проверка работ проводилась централизованно и по единым критериям, призеры и победители определялись для каждого класса отдельно. Задания олимпиады были подготовлены для двух возрастных категорий (9–10-е и 11-е классы) в нескольких равноценных вариантах. Ниже приводятся условия и решения одной из задач каждого типа.

Межрегиональная олимпиада школьников по информатике и компьютерной безопасности на основании предложения Российского совета олимпиад школьников включена в Перечень олимпиад

школьников на 2015/16 учебный год (профиль — информатика, уровень — 3).

Условия и решения задач

Задача 1 (9–10-е, 11-е классы)

Блок-схема алгоритма обработки строки *STR* изображена на *рис.* 1. Функция *length*, как и обычно, возвращает количество символов в строке. Чему была равна переменная *STR* перед началом выполнения алгоритма, если при $N = 10$ в результате обработки было получено значение переменной *STR* = ‘*bdieghtreen tre*’?

Решение

Введем следующие обозначения:

N — количество шагов алгоритма;

m — количество символов в строке;

STR — строка.

Из анализа блок-схемы следует, что данный алгоритм реализует перестановку символов в исходной строке *STR* следующим образом. Алгоритм состоит из десяти итераций. На каждой итерации алгоритма последовательно просматривается исходная строка *STR*. При этом ее содержимое изменяется по следующим правилам:

Случай 1. Номер итерации алгоритма *k* четный.

В этом случае каждый символ строки, имеющий четный индекс *i*, последовательно меняется местами с элементом, имеющим индекс $i + 2$.

Случай 2. Номер итерации алгоритма нечетный.

В этом случае каждый символ строки последовательно меняется местами с каждым следующим элементом строки, причем последний обмен происходит в паре STR[m = 2], STR [m – 1], а не в последней паре.

Обозначим через B = (b, d, i, e, g, h, t, r, e, e, n, ‘‘, t, r, e) упорядоченный набор символов, соответствующий содержимому переменной *STR* после работы алгоритма, описанного блок-схемой, при $N = 10$.

Обозначим через A = (1,2,3,4,5,6,7, 8,9,A,B,C,D,E,F) упорядоченный набор символов, отличных от элементов, входящих в B.

Пусть на вход подается строка *STR*, содержимое которой соответствует упорядоченному набору A. Тогда после обработки строки *STR* с использованием заданного алгоритма при $N = 10$ ее содержимое с одной стороны описывается перестановкой

$C = (A,D,C,1,E,3,2,5,4,7,6,9,8,B,F)$

элементов упорядоченного набора A и совпадает с упорядоченным набором B с другой стороны, т.е.

$C = (A,D,C,1,E,3,2,5,4,7,6,9,8,B,F) =$

$= (b, d, i, e, g, h, t, r, e, e, n, ‘‘, t, r, e) = B.$

Сопоставив соответствующие символы этих упорядоченных наборов, получаем, что в исходном наборе $1 = e, 2 = t, 3 = h, 4 = e, 5 = r, 6 = n, 7 = e, 8 = t, 9 = ‘‘, A = b, B = r, C = i, D = d, E = g, F = e$, следовательно, на вход алгоритма изначально подавалась строка ***ethernet bridge***.

Ответ: ethernet bridge.

Задача 2 (9–10-е, 11-е классы)

Вводимые пользователями пароли преобразуются с помощью функции, исходный код на Си которой представлен ниже. Для проверки правильности пароля возвращаемое функцией значение сверяется с эталоном, хранящимся в базе данных. Известно, что для некоторого логина установлен пароль “БЕЗОПАСНОСТЬ”. Предложите пароль минимальной длины такой, что он также успешно пройдет проверку.

```
int HASH(char *text)
{
   int k = 0, H;                          //1
   char temp[6], letter = 'G';            //2
   for(;(k < strlen(text))&&(k < 6);++k)
       temp[k] = text[k];                 //3
   if (k < 6) for(;k < 6;k++)
        temp[k] = letter;                 //4
   H = ((temp[0] + temp[1] +
        temp[2])&7) * 64;                 //5
   H += ((temp[0] + temp[2] +
        temp[4])&7) * 8;                  //6
   H += ((temp[1] + temp[3] +
        temp[5])&7);                      //7
       return H;                          //8
}
```

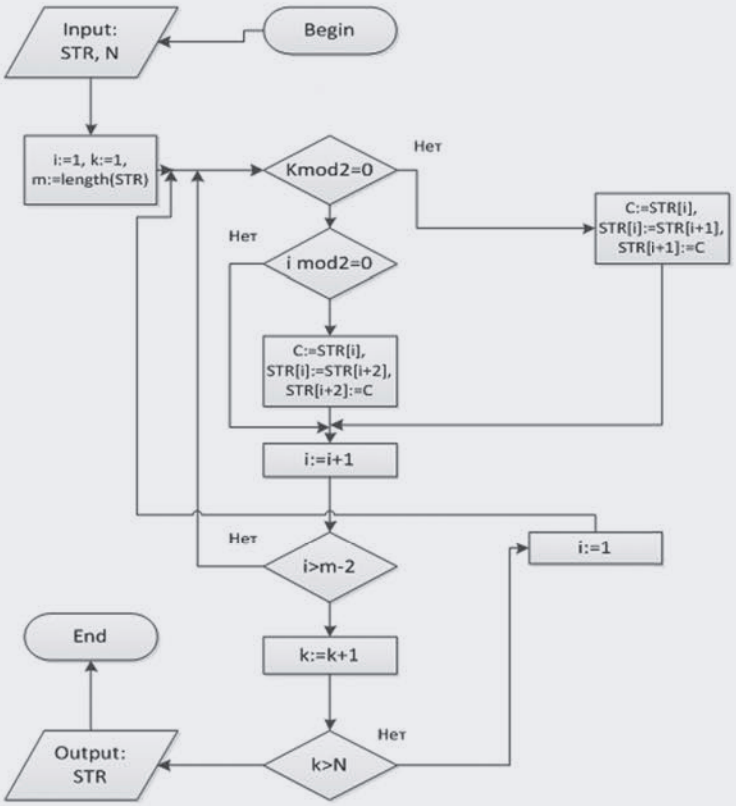


Рис. 1. Блок-схема алгоритма обработки строки *STR*

Решение

В программе использованы следующие переменные:

text — входной параметр функции, в которой передается строка, для которой будет вычислено значение функции;

k — вспомогательная переменная;

temp — символьный массив из шести элементов, используемый для хранения промежуточных данных при вычислениях;

H — целое число типа *int* (для хранения результата используются младшие девять бит).

В цикле, реализованном в строке 3, происходит выборка первых шести символов из входной строки. В строке 4 происходит проверка на длину строки *temp*. Если полученная в результате работы предыдущего цикла строка меньше шести, то она дополняется символами “G”. Это преобразование существенно при подборе пароля. Таким образом, после выполнения этой части программы (с учетом того, что по условию задачи преобразуется пароль “КОНФИДЕНЦИАЛЬНОСТЬ”) переменная *temp* будет иметь заполнение “КОНФИД”.

В строке с номером 5 начинает формироваться выходное значение: *ASCII*-коды нулевого, первого и второго символов буфера *temp* складываются (по модулю 8), а результат записывается в 6, 7, 8-й биты переменной *H*. В результате выполнения строки шесть аналогично заполняются 3, 4, 5-й биты *H*, а в строке семь — биты с номерами 0, 1, 2. Таким образом, после выполнения функции, в которой преобразуется пароль “КОНФИДЕНЦИАЛЬНОСТЬ”, возвращаемый результат равен $382_{10} = 101\ 111\ 110_2$.

Для решения задачи необходимо предложить такой пароль, образ которого, полученный в результате работы функции, будет равен 382. Заметим, что из-за процедуры составления строки *temp*, описанной выше, достаточно рассмотреть только пароли длиной менее семи символов.

Запишем искомый пароль в виде $a_1\ a_2\ a_3\ a_4\ a_5\ a_6$. Учитывая описанный алгоритм построения образа и представление в двоичном виде образа пароля “КОНФИДЕНЦИАЛЬНОСТЬ”, запишем систему уравнений, которой должен удовлетворять искомый пароль:

$$\begin{cases} a_1 + a_2 + a_3 = 5 \pmod 8 \\ a_1 + a_3 + a_5 = 7 \pmod 8 \\ a_2 + a_4 + a_6 = 6 \pmod 8. \end{cases}$$

Решение будем проводить перебором по числу символов.

Предположим, что длина искомого пароля 0 или 1. Тогда третье уравнение системы принимает вид 5 = 8 (mod 8). Таким образом, длина пароля должна быть больше 1.

Пусть длина искомого пароля равна двум, тогда `temp[6] = a1a2"GGGG"` и система принимает вид:

$$\begin{cases} a_1 + a_2 + 71 = 5 \pmod 8 \\ a_1 + 2 * 71 = 7 \pmod 8 \\ a_2 + 2 * 71 = 6 \pmod 8). \end{cases}$$

Данная система несовместна, а значит, длина пароля больше двух. Аналогичный результат получается при рассмотрении пароля из трех символов (`temp[6] = "a1a2a3GGGG"`).

Пусть длина искомого пароля равна четырем. Тогда `temp[6] = "a1a2a3a4GGGG"` и система принимает вид:

$$\begin{cases} a_1 + a_2 + a_3 = 5 \pmod 8 & (1) \\ a_1 + a_3 + 71 = 7 \pmod 8 & (2) \\ a_2 + a_4 + 71 = 6 \pmod 8). & (3) \end{cases}$$

Вычитая из уравнения (1) уравнение (2), получаем $a_2 = 5$, подставляя это значение в уравнение (3), получаем $a_4 = 2$. А из уравнения (2) имеем, например, $a_1 = 3$, $a_3 = 5$. Таким образом, один из искомых паролей равен CEEB.

Ответ: CEEB (в латинском алфавите).

Задача 3 (11-й класс)

В штате секретной службы состоят десять агентов (под номерами 1, 2, …, 10). Для связи с ними при проведении разведывательной операции используются устройства, которые работают в заданном диапазоне частот, но в них можно настроить индивидуально интенсивность передачи сигнала в минуту (число сигналов в минуту). В случае провала агент отключает передатчик. В штабе стоит приемное устройство, которое считает общее количество пришедших в минуту сигналов от всех агентов. Как надо задать частоты передатчиков, чтобы в штабе в случае провалов агентов можно было бы определить их номера?

Решение

В штабе каждую минуту получают информацию о суммарном числе сигналов *N*. Представим *N* в виде сумм степеней двойки

$$N = a_{n-1}\,2^{n-1} + a_{n-2}\,2^{n-2} + \ldots + a_1 2 + a_0.$$

В силу свойств позиционных систем счисления коэффициенты a_{n-1}, a_{n-2}, … a_1, a_0 определяются однозначно для *N*. Если агент с номером *i* настроит свой передатчик на передачу 2^{i-1} символов в минуту, то по коэффициентам представления суммарного числа сигналов *N* в двоичной системе счисления легко можно определить действующих и провалившихся агентов. Если коэффициент a_j равен 0, то агент провалился, если 1 — действует.

Задача 4 (11-й класс)

Известно, что в текстовом файле скрыто сообщение. Для его внедрения использовался регистр символа: буква в нижнем регистре соответствует “0”, в заглавном регистре — “1”. Сообщение представляется восьмиразрядными кодами символов (ASCII), составленными из последовательно считанных бит. Для внедрения используются только буквы русского и английского алфавита. Реализуйте приложение, извлекающее скрытое сообщение из файла.

Указание

Приложение разрабатывается на базе реализованного шаблона, находящегося в папке с заданием.

Для получения текста из файла необходимо вызвать функцию:

```
void GetStegoText(char *massiv,
                  int *resultlen);
```

massiv — указатель на массив символов, который будет заполнен сообщением после возврата из функции (не менее 500 байт);

resultlen — указатель на целочисленную переменную, которая будет равна количеству записанных в *massiv* во время выполнения функции байт.

Для проверки, является ли символ буквой, используйте функцию:

```
int iswalpha(unsigned char c);
```

Для перевода символов в верхний и нижний регистр используйте функции:

```
int toupper(unsigned char c);
int tolower(unsigned char c);
```

Для корректной работы строки, содержащие русские буквы, должны быть объявлены как `unsigned char`.

Заметим, что если вы работаете со средой обработки *Visual Studio*, то необходимо обратить внимание на настройку проекта (правой кнопкой на проекте "Свойства"): параметр "*Character Set*" должен быть установлен в "*Use Multi-Byte Character Set*".

Решение

Переберем все элементы массива кодированных символов и запишем в новый массив "1", если символ в верхнем регистре, иначе запишем "0".

```
char *openmass = new char[size + 1];
for(int i = 0;i < size;i++)
{
   if (toupper(cryptomass[i]) ==
       cryptomass[i])
       openmass[i]| = 1;
   else
       openmass[i] = 0;
}
```

Наложим ограничивающее условие "Знаки препинания и цифры не учитываются", что производится установкой блокирующего условия или цикла, пропускающего очередной символ кодированного массива в случае, если он является знаком препинания или цифрой:

```
while(!iswalpha((unsigned char)
      cryptomass[i]) && i < size) i++;
```

Таким образом, получаем рабочий цикл, переводящий массив кодированных байтов в массив из нулей и единиц, представляющих собой символы русского алфавита, записанные в двоичном представлении. Последним действием по конструированию алгоритма и написанию программы является запись открытого текста в виде привычных символов, что производится с помощью сдвиговых операций. Необходимо пройти выходной массив один раз от начала до конца, записывая нули и единицы по восемь элементов в один байт с использованием, к примеру, вот такого выражения:

```
openmass[l] |= 1<<(7 - j);
```

Здесь *j* — это инкрементная переменная, принимающая значения от 0 до 7 и позволяющая с помощью операции сдвига сместить двоичную единицу последовательно по всем возможным значениям в байте (1, 2, 4, 8, 16, 32, 64, 128). Слева от равенства используется знак поразрядного сложения, задача которого — установить в результирующем байте декодированной последовательности битовую единицу на соответствующее место (см. *рис.* 2).

Задача решена, но для максимальной оценки необходимо провести оптимизацию, чтобы устранить лишнее преобразование из массива байтов в массив нулей и единиц, а затем обратно в массив байт. Получается, что входной массив преобразуется в меньший в восемь раз по объему (если не учитывать знаки пунктуации и цифры).

Значит, необходимо создать выходной массив сразу и заполнять его последовательно в байтовом представлении. Рабочий цикл приведен ниже в листинге.

```
int _tmain(int argc, _TCHAR* argv[]){
     setlocale(LC_ALL,"Russian");
     unsigned char cryptomass[5000];
     int cryptomasssize=0;
//////////////////////////////////////
     GetCryptoText(cryptomass,
                   &cryptomasssize);
      //Запишите свое решение ниже
      //...
      char *openmass;
     int size = strlen((char*)cryptomass);
      unsigned char mask = 1;
      int k = 0, l = 0, t = 0;
      openmass = new char[size/8 + 1];
      for(int i = 0;i < size;i++) {
      openmass[l] = 0;
       for(int j = 0;j < 8;j++) {
           while(!iswalpha((unsigned char)
                cryptomass[i]) && i < size)
                i++;
            if (toupper(cryptomass[i]) ==
                cryptomass[i])
               openmass[l] |= 1 << (7 - j);
                 i++;
```

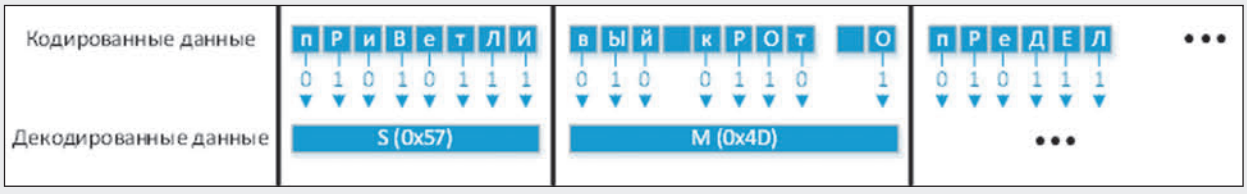


Рис. 2. Сдвиговые операции

```
        }
        l++;
        i--;
      }
      openmass[size/8]='\0';
      printf("%s\n",openmass);
////////////////////////////////////
      return 0;
}
```

Задача 5 (9–10-е классы)

Агент секретной службы Смит — начальник отдела по сбору секретных и компрометирующих материалов, в котором для хранения информации создан специальный закрытый ресурс *PiggyLeaks.ru*, доступ к которому осуществляется с использованием пароля. Каждому сотруднику отдела выдается свой уникальный пароль доступа к материалам. При генерации паролей агент Смит ввел следующие ограничения:

• пароль состоит из десяти цифр, каждая из которых может принимать значение от 1 до 6 включительно;

• сумма любых трех соседних цифр в пароле равна десяти.

Помогите агенту Смиту написать программу по генерации паролей для своих сотрудников. Сколько всего сотрудников может работать у него в отделе?

Решение

Представим последовательность цифр в виде:

$$x_1x_2x_3x_4x_5x_6x_7x_8x_9x_{10}.$$

Покажем, что в последовательности будут повторяться первые три цифры.

Рассмотрим первую тройку цифр: $x_1x_2x_3$. Зафиксируем x_1 и x_2. Тогда по условию $x_3 = 10 - x_1 - x_2$.

Рассмотрим вторую тройку цифр: $x_2x_3x_4$.

x_2 и x_3 подставим из предыдущего шага. Найдем x_4:

$x_4 = 10 - x_2 - x_3 = 10 - x_2 - (10 - x_1 - x_2) = x_1$.

Аналогично x_5:

$x_5 = 10 - x_3 - x_4 = 10 - (10 - x_1 - x_2) - x_1 = x_2$.

В итоге получим последовательность вида

$$x_1x_2x_3x_1x_2x_3x_1x_2x_3x_1.$$

Задача сводится к перебору x_1 и x_2 таких, что $10 - x_1 - x_2 \geq 1$ и $10 - x_1 - x_2 \leq 6$.

Это реализуется двумя вложенными циклами от 1 до 6 включительно.

```
#include <stdio.h>
#include <iostream>
using namespace std;
int main() {
  int i, j, k;
  int total = 0;
  for (i = 1; i <= 6; i++)
  {
    for (j = 1; j <= 6; j++)
    {
      k = 10 - i - j;
      if (k <= 6 && k >= 1)
      {
        cout << i << j << k << i << j <<
            k << i << j << k << i << endl;
        total++;
      }
    }
  }
  cout << endl << " Всего: " << total
       << "комбинаций" << endl;
  return 0;
}
```

В результате работы программы переменная *total* будет содержать количество сотрудников, которые могут работать у Смита и иметь различные пароли.

Ответ: 27 сотрудников.

Задача 6 (9–10-е классы)

На волшебной горе расположена система озер и рек. Верхнее озеро образуется из тающего ледника. Из каждого озера вытекают две реки, а впадает одна (за исключением верхнего). Первым на горе возле верхнего озера поселился гном Тим. Каждый новый обитатель горы строил свой дом на одном из озер как можно ближе к Тиму. На каждом озере селился только один гном. Сейчас на горе живут 2014 гномов. Какое количество рек надо проплыть Тиму, чтобы попасть в гости к последнему поселившемуся гному?

Решение

Схему волшебного города можно представить в виде двоичного дерева. В качестве узлов рассматриваются озера с построенными на них домами, в качестве входящих и исходящих ребер — протоки. Корнем дерева (выделенной вершиной) является озеро с домом Тима. Очевидно, что в двоичном дереве число узлов, расстояние от корня до которых равно F, не превосходит 2^F. На расстоянии 1 может находиться не более двух узлов, на расстоянии 2 — не более четырех узлов и т.д. В таблице приведены значения максимального количества домов на расстоянии F:

Расстояние от корня F	Число домов 2^F
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048

Правила застройки города гарантируют появление нового дома на расстоянии $F+1$ в том и только в том случае, если все возможные места на расстоянии F уже заполнены. При этом число домов,

находящихся не далее, чем на расстоянии F, равно $1 + 2 + 4 + \ldots + 2^F = 2^{F+1} - 1$.

В нашем случае могут быть вакантные места для строительства на расстоянии F, поэтому должно выполняться неравенство $2^{F+1} - 1 \geq 2014$. Очевидно, что при $F = 9$ выполнено неравенство $2^{10} - 1 < 2014$, а при $F = 10$ — неравенство $2^{11} - 1 \geq 2014$. Таким образом, максимальное количество проток, которое надо проплыть Тиму до самого дальнего дома, равно 10.

Ответ: 10 проток.

Задача 7 (11-й класс)

Иван написал Егору закодированное сообщение, используя следующую таблицу:

№	Символ	Код	№	Символ	Код
1	*а*	010	18	*р*	001
2	*б*	000	19	*с*	110001
3	*в*	011000	20	*т*	110010
4	*г*	011001	21	*у*	110011
5	*д*	101	22	*ф*	110100
6	*е*	011010	23	*х*	110101
7	*ё*	011011	24	*ц*	110110
8	*ж*	0111000	25	*ч*	110111
9	*з*	0111001	26	*ш*	111000
10	*и*	0111010	27	*щ*	111001
11	*й*	0111011	28	*ъ*	111010
12	*к*	100	29	*ы*	111011
13	*л*	0111100	30	*ь*	111100
14	*м*	0111101	31	*э*	111101
15	*н*	0111110	32	*ю*	111110
16	*о*	0111111	33	*я*	111111
17	*п*	110000			

В результате была получена последовательность битов открытого текста $O(i)$, $i = 1, \ldots, 33$. Затем он произвел преобразование

$S(i) = (O(i) + S(i-1)) \bmod 2$, $S(0) = 0$ (*рис.* 3).

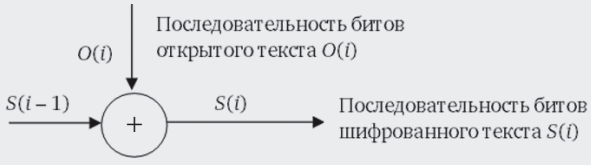


Рис. 3. Схема преобразования закодированного сообщения

В результате была получена последовательность битов шифрованного текста $S(i)$, $i = 0, \ldots, 33$:

00111111100110000110011000000001100.

Какое сообщение Иван написал Егору?

Решение

Согласно схеме шифрования i-й бит исходного открытого сообщения $O(i)$ равен сумме $(S(i) + S(i-1)) \bmod 2$ для $i > 0$. Учитывая, что $S(0) = 0$ и $S(1) = O(1)$, получаем последовательность битов открытого сообщения $i = 1, \ldots, 33$

010000001010100010101010000001010.

Воспользуемся таблицей кодов. Ни один символ не закодирован 0 и 01. Последовательность битов 010 соответствует символу ***а***. Последующие три бита 000 — символу ***б***. Рассуждая аналогично, получаем:

010 — ***а***; 000 — ***б***; 001 — ***р***; 010 — ***а***; 100 — ***к***;

010 — ***а***; 101 — ***д***; 010 — ***а***; 000 — ***б***;

001 — ***р***; 010 — ***а***.

Таким образом, было закодировано слово *абракадабра*.

Ответ: абракадабра.

Задача 8 (9–10-е классы)

Что делает программа на языке *Pascal*?

```
                Uses crt;
               Const   z=10;
  hhh=10; y=         2*
hhh+    1;s=    {}  hhh-       z+
1;          Type     vec={a  -1  ab
x}     {} array[s..z]  {}  of  {a  ,v
v+1}     integer;        {}  var a,b:{}
  vec;v,f:{hh}        {xy}     {ab}
  integer;          x:Boolean;{}
  Begin clrscr;{a,b,x}  {xy}
    {ab}  Randomize;     {}
{hh}      {}    write(      '[]='          );for  v:=
  s         to               z do{v  x}     begin//
 f:=    random               (y);a[v]:=  hhh-f+s-1;
Write(a[v]:3);End;           writeln;b:= a;Repeat{}
  x:={x}     true;for{a     }v:=s to   z-  1 do if{}
  b[  v]       >b[v+1]then Begin{z}  f:=b[v];b[
  v]  :=  b[     v+1];b[v+1]:=f;{}    x:=false;
        End;     until x;{ab          }write(
      '[]=');for v:=s                  to
      {v-1}z
        do{v+1-2*x}
        write{x[a-2]
          }(b[v]:3
          );  End.
```

Решение

1. Как известно, в *Pascal* комментарии могут быть обозначены в фигурных скобках и начинаться со строки “//”. Удалим комментарии и осуществим правильные переносы строк в исходном тексте. Получим:

```
Uses crt;
Const z = 10;
hhh = 10;
y = 2 * hhh + 1;
s = hhh – z + 1;
Type vec = array[s..z] of integer;
var a, b: vec;
v, f: integer;
x: Boolean;

Begin
ClrScr;
Randomize;

Write('[]=');
for v := s to z do
begin
    f := random(y);
    a[v] := hhh – f + s – 1;
    Write(a[v]: 3);
End;
writeln;

b := a;

Repeat
      x := true;
```

```
      for v := s to z – 1 do
      if b[v] > b[v + 1] then
      Begin
           f := b[v];
           b[v] := b[v + 1];
           b[v + 1] := f;
           x := false;
      End;
Until x;
write('[]=');
for v := s to z
    do Write(b[v]: 3);
End.
```

2. Избавимся от запутывающих лишних переменных в начале программы:

```
Uses crt;
Const z = 10;
hhh = 10;
y = 21;
s = 1;
Type vec = array[1..z] of integer;
var a, b: vec;
v, f: integer;
x: Boolean;
```

3. Заменим фиктивные переменные числами и объединим переменные с одинаковыми значениями:

```
Uses crt;
Const z = 10;
y = 21;
Type vec = array[1..z] of integer;
var a, b: vec;
v, f: integer;
x: Boolean;

Begin
ClrScr;
Randomize;
Write('[]=');
for v := 1 to z do
begin
    f := random(y);
    a[v] := z – f ;
    Write(a[v]: 3);
End;
writeln;
b := a;
Repeat
      x := true;
      for v := 1 to z – 1 do
      if b[v] > b[v + 1] then
      Begin
           f := b[v];
           b[v] := b[v + 1];
           b[v + 1] := f;
           x := false;
      End;
Until x;
write('[]=');
for v := 1 to z
    do Write(b[v]: 3);
End.
```

4. Переименуем переменные по смыслу и пронумеруем строки

```
Uses crt;     //1
Const length = 10;                        //2
range = 21;                               //3
Type vector = array[1..length] of
     integer;                             //4
var a, b: vector;                         //5
index, number: integer;                   //6
flag: Boolean;                            //7
Begin                                     //8
ClrScr;                                   //9
Randomize;                               //10
Write('[]=');                            //11
for index := 1 to length do              //12
begin                                    //13
    number := random(range);             //14
    a[index] := length – number;         //15
    Write(a[index]: 3);                  //16
End;                                     //17
writeln;                                 //18
b := a;                                  //19
Repeat                                   //20
  flag := true;                          //21
  for index := 1 to length – 1 do        //22
  if b[index] > b[index + 1] then        //23
  Begin                                  //24
     number := b[index];                 //25
     b[index] := b[index + 1];           //26
     b[index + 1] := number;             //27
     flag := false;                      //28
      End;                               //29
Until flag;                              //30
write('[]=');                            //31
for index := 1 to length
    do Write(b[index]: 3);               //32
End.                                     //33
```

5. Разбираем код на смысловые части:

• в строках с 1-й по 7-ю происходит инициализация переменных;

• в строках с 8-й по 19-ю заполняется массив длины 10 числами в диапазоне от –10 до 10 и выводится на экран;

• в строках с 20-й по 30-ю сортируется массив методом “пузырька”;

• в строках с 31-й по 33-ю получившийся массив выводится на экран.

Ответ: программа реализует алгоритм сортировки массива методом “пузырька”.

С задачами прошедших олимпиад по информатике и компьютерной безопасности и их решениями можно также ознакомиться на сайте *http://www.v-olymp.ru*.

Педагогический университет
«Первое сентября»

Лицензия Департамента образования г. Москвы 77Л01 № 0007183, рег. № 036377 от 23.07.2015

ДИСТАНЦИОННЫЕ КУРСЫ ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

(с учетом требований ФГОС)

До 15 января 2016 г. ведется прием заявок на второй поток 2015/16 учебного года

образовательные программы:

- НОРМАТИВНЫЙ СРОК ОСВОЕНИЯ – 108 УЧЕБНЫХ ЧАСОВ

  Стоимость – 4990 руб.

- НОРМАТИВНЫЙ СРОК ОСВОЕНИЯ – 72 УЧЕБНЫХ ЧАСА

  Стоимость – от 3990 руб.

По окончании выдается удостоверение о повышении квалификации установленного образца

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

Пожалуйста, обратите внимание:

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

ИСТОРИЯ ИНФОРМАТИКИ

Все ли изобретено?

В.В. Шилов,
Москва

"Знаете вы это или нет, но в начале XX века человечество было уверено, что «все уже изобретено и дальше прогрессу двигаться некуда». Да-да!".

Вот такое высказывание прочитал я как-то раз в блоге одного компьютерного эксперта. Нет-нет, сказал я сам себе, не знаю! Высказывание меня заинтересовало, и спустя минуту-другую я читал, уже на другом ресурсе: "Думаю, вы знаете историю о Чарлзе Дуэле, сотруднике Патентного ведомства США, который в 1899 году произнес легендарную фразу: «Все, что можно было изобрести, уже изобретено». Но вас может удивить и тот факт, что такие люди встречаются и поныне"[1].

Нет, снова вынужден был признать я, не знаю. Я даже не знаю, кто такой Чарлз Дуэл, и, соответственно, стоит ли придавать серьезное значение его, оказывается, "легендарной" фразе. Мало ли кто и мало ли что говорит, не подумав, в полемическом запале или попросту по глупости. А уж от какого-то никому не ведомого "сотрудника ведомства" ожидать жемчужин мудрости и вовсе не приходится!

Чарльз Дьюэлл

Увы, в русском сегменте Интернета никакой дополнительной информации по заинтересовавшему меня вопросу я не обнаружил — блогеры, "ИТ-эксперты" и прочие добросовестно переписывали друг у друга приведенное выше высказывание и в результате растиражировали его десятками тысяч ссылок. Правда, насторожило меня то, что на некоторых сайтах оно соседствовало с десятками других приписываемых разным людям высказываний, — из которых как минимум некоторые приписываются им заведомо ошибочно.

Поскольку всегда лучше всего обращаться к первоисточникам, то поиск я продолжил уже в англоязычном сегменте Интернета. Информации о Чарльзе Дьюэлле — а именно так надо правильно писать его имя — и там оказалось весьма и весьма немного, даже основные биографические сведения обнаружить удалось далеко не сразу. Но то, что из этой биографии стало известно, настраивало скорее на положительный лад.

Дьюэлл родился в 1850 г. в штате Нью-Йорк в семье юриста и по примеру отца выбрал эту профессию. В возрасте 22 лет он получил степень бакалавра права и занялся юридической практикой. Необходимо сказать, что Дьюэлл специализировался в области патентного права; здесь ему, несомненно, пригодился опыт, полученный во время двухлетней работы (1875–1876 гг.) в качестве эксперта Бюро патентов США. В течение многих лет он был одним из партнеров юридической фирмы, занимавшейся этими вопросами. Дьюэлл принимал участие в самых сложных и запутанных судебных процессах своего времени, связанных с патентными спорами. В 1899 г. Чарльз Дьюэлл опубликовал серьезное академическое исследование различных аспектов организации и функционирования патентной системы.

Не приходится сомневаться, что именно накопленный Дьюэллом колоссальный опыт в сочетании с высокой квалификацией в области патентного права побудили президента Мак-Кинли назначить его в 1898 г. в Бюро патентов США — и не рядовым "сотрудником патентного ведомства", а его руководителем (Commissioner of Patents). Еще одним основанием для назначения могло стать и то, что Дьюэлл был активистом республиканской партии и, начиная с 1878 г., несколько раз избирался в законодательное собрание штата Нью-Йорк.

Вероятно, деятельность Дьюэлла (а он возглавлял Бюро патентов в течение трех лет, до 1901 года) на этом ответственном посту получила высокую оценку. Об этом можно судить по тому, как успешно складывалась его дальнейшая полити-

[1] Меня очень удивило само построение фразы, было непонятно, какие "такие люди встречаются и поныне"? Сотрудники Патентного ведомства? Знающие историю о Чарлзе Дуэле? Или произносящие легендарные фразы? — **Прим. авт.**

ческая карьера. Известно, что в 1905–1907 гг. он был членом апелляционного суда[2] федерального округа Колумбия, а во время президентских выборов 1908 года — членом коллегии выборщиков от штата Нью-Йорк. Чарльз Холланд Дьюэлл скончался в 1920 году, не дожив двух месяцев до своего семидесятилетия.

Итак, поверить в то, что этот человек, профессионал, прекрасно знакомый с механизмами и состоянием патентного дела в США, мог сморозить столь очевидную глупость, затруднительно. Но, собственно говоря, верить и не надо. Американские исследователи, соотечественники Дьюэлла, уже давно выяснили, что утка о его "высказывании" была запущена некими журналистами, издавшими в 1981 г. книгу, из которой до сих пор черпают непроверенную и недостоверную информацию многие любители сенсаций.

Конрад Цузе (фото 1991 г.)

Причем в своей книге они не только приписали Дьюэллу это самое высказывание — которое, похоже, сами же и придумали! — но и уверяли, будто он еще и обратился к президенту США с предложением ликвидировать за ненадобностью само Бюро патентов! Нелепость этих утверждений становится очевидной, если прочитать написанный Дьюэллом и как раз адресованный президенту страны отчет о деятельности Бюро патентов за 1899 год (год, которым все дружно датируют якобы принадлежащее ему высказывание). Так, Дьюэлл указывает, что количество выданных в этом году патентов увеличилось по сравнению с предыдущим на целых три тысячи — вряд ли он мог полагать, что все изобретено... Кроме того, он заявляет, что "мы должны сделать изобретения одним из самых мощных орудий достижения прогресса и процветания нации". Тезис, странный для человека, который одновременно просит президента распустить патентное ведомство!

Интересно, что на неоднократные обращения специалистов к авторам книги и опубликовавшему ее издательству с просьбой указать источник, откуда были заимствованы приписываемые Дьюэллу слова, ответ так ни разу и не был получен…

Итак, мы разобрались — разумеется, Чарльз Дьюэлл не произносил и не писал того, что ему приписывают.

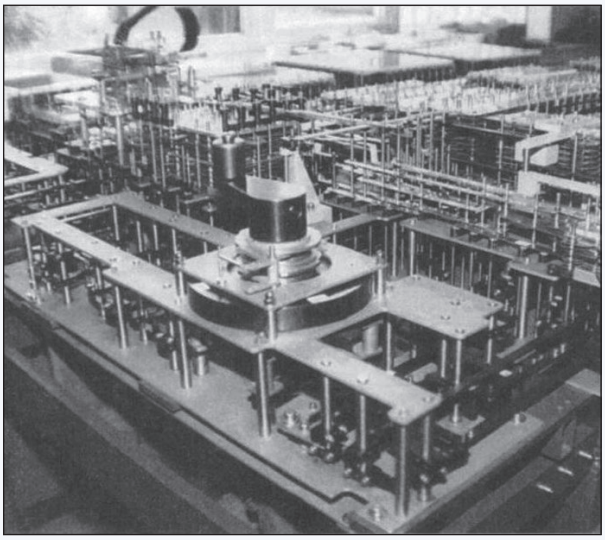
Однако справедливости ради стоит сказать, что сходные по тональности высказывания время от времени действительно звучали. Так, в написанном в 1913 г. романе великого британского фантаста Герберта Уэллса "Освобожденный мир" мы читаем: "Все великие открытия уже сделаны, — писал Джеральд Браун в своем обзоре девятнадцатого столетия. — Нам остается лишь разрабатывать кое-какие детали"[3].

Однако это высказывание не получило столь же широкой известности: во всяком случае, никаких ссылок на него, помимо книги Уэллса, мне найти не удалось. Не нашел я ни его источника, ни сведений о его авторе, Джеральде Брауне…

Но до сих пор речь у нас шла об изобретениях и открытиях "вообще", — так сказать, в философском плане. А если говорить о какой-либо конкретной науке или предметной области, то такое мнение (а точнее, заблуждение) иногда действительно овладевает умами специалистов в самих этих науках и предметных областях.

Приведу только один пример. Сегодня представить нашу жизнь без компьютеров совершенно невозможно. Но хотя сама идея управления работой вычислительной машины посредством программ была высказана около 1840 г. великим английским ученым Чарльзом Бэббиджем, воплотилась в жизнь она только сто лет спустя. Споры о том, кто же сумел сделать это первым, не прекращаются на протяжении многих лет, но с тем, что выдающийся немецкий инженер и изобретатель Конрад Цузе (1910–1995) был одним из первых, соглашаются все историки без исключения.

Цузе начал работать над проектом механической вычислительной машины еще в 1936 г. Основными идеями, которые он сформулировал и собирался реализовать, были: полная автоматизация процесса вычислений, управляемого считываемой с перфоленты последовательностью команд; использование двоичной системы; наличие устройства памяти большой емкости; выполнение операций над числами с плавающей точкой. Более того, в том же году у Цузе оформилась идея хранимой в памяти программы, — едва ли не центральная идея в организации современных компьютеров.



Вычислительная машина Цузе модели Z1 (фото 1941 г.)

Много лет спустя в своих интереснейших мемуарах Цузе написал: "Был 1937 год, и наша работа

[2] Высшая судебная инстанция в ряде штатов США.

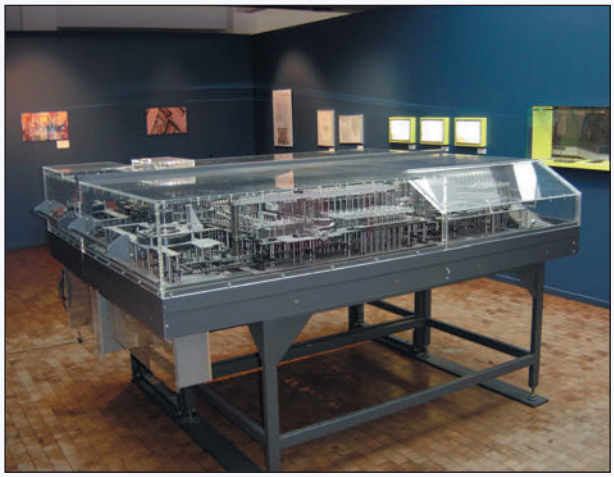
[3] *Уэллс Г.* Собрание сочинений в 15 тт. Т. 4. М.: Правда, 1965. С. 308.

достигла той стадии, когда уже потребовалось лучшее финансирование. Я связался с изготовителем счетных машин (*имеются в виду арифмометры*. — **В.Ш.**) доктором Куртом Паннке… «Мне сказали, — начал доктор Паннке, — что вы изобрели счетную машину. Я не хочу отговаривать вас продолжать изобретать и развивать новые идеи, но должен со всей ответственностью заявить: в области счетных машин практически уже все и во всех подробностях исследовано и усовершенствовано. Едва ли осталось что-то не изобретенное; знаменитый конструктор счетных машин Гаманн — было изготовлено около миллиона счетных машин его конструкции — подтвердил мне это»”[4].

Следует признать, что в чем-то доктор Паннке был прав: выпускавшиеся в те времена механические арифмометры действительно были близки к совершенству. И Кристель Гаманн тоже был по-своему прав, подтверждая его слова. Ведь именно он как раз и был автором последних усовершенствований в трехвековой истории механических счетных машин — арифмометры выпускали вплоть до 70-х годов прошлого века, однако ничего принципиально нового в их конструкции после Гаманна больше так и не появилось[5]. Другое дело, что ни доктор Паннке, ни Гаманн не учли возможность появления вычислительных машин, построенных на совершенно новых принципах — в частности, на принципе программного управления!

Арифмометр К.Гамана “Mercedes Euclid”

Тем не менее доктор Паннке предоставил в распоряжение молодого изобретателя необходимые средства. Благодаря этому через год Цузе сумел завершить работу над своей первой, полностью механической, вычислительной машиной, получившей название Z1. И хотя она работала еще не слишком надежно, эксперименты убедили Цузе в правильности главных идей ее архитектурной организации. С 1989 г. восстановленная под наблюдением и при участии самого Конрада Цузе копия машины Z1 экспонируется в Берлинском музее транспорта и технологии.



Восстановленная модель Z1 в музее

Поэтому странно сегодня слышать заявления некоторых экспертов, будто развитие вычислительной техники прекратилось, что наблюдается дефицит идей или что дальнейшее совершенствование компьютерных архитектур невозможно. Думаю, что жизнь очень скоро опровергнет все эти пессимистические прогнозы.

Я уверен, что изобретено и открыто еще далеко не все. И если не в нашей власти запретить падким до сенсаций журналистам совершать свои “открытия”, то по крайней мере мы вполне можем не доверять слепо их авторам и уж, во всяком случае, не тиражировать их измышления.

GAMES.EXE

Игра “Две кучки камней”

Пятачок и Винни-Пух играют в такую игру. Перед ними лежат две кучки камней, по 15 штук в каждой. Игроки по очереди берут любое количество камней из любой кучки (но не из двух кучек сразу). Проигрывает тот, кто не сможет в свою очередь взять камни. Кто выиграет в эту игру — начинающий или делающий ход вторым? Почему?

[4] *Zuse K.* The Computer — My Life. N.-Y.: Springer Verlag, 1993. P. 42.

[5] Известны четыре принципиально различные конструкции арифмометров. Одна была предложена в конце XVII в. гениальным немецким ученым Готфридом Вильгельмом Лейбницем, вторая — в конце XIX в. практически одновременно Вильгодтом Однером в России и Фрэнком Болдуином в США, а третья и четвертая — в начале XX столетия в Германии Кристелем Гаманном.

“ЛОМАЕМ” ГОЛОВУ

Странный мальчик

Один странный мальчик говорит правду только по средам и пятницам, по вторникам всегда лжет, а в остальные дни может и солгать, и сказать правду. Семь дней мальчика спрашивали, как его зовут. Первые шесть ответов, по порядку, были таковы: Женя, Боря, Вася, Вася, Петя, Боря. Можно ли узнать, как на самом деле зовут этого мальчика? Как он ответит на седьмой день?

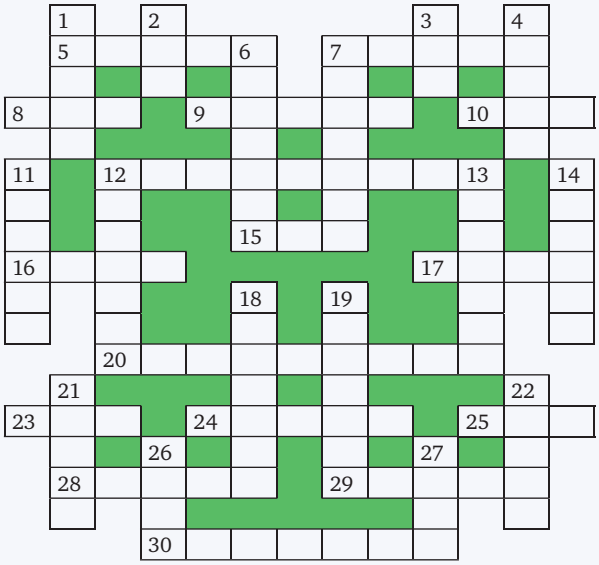
Ночной переход по мосту

Однажды ночью семья из шести человек подошла к мосту. Мальчик может перейти его за 4 минуты, его сестра — за 6, папа — за 1, мама — за 3, дедушка — за 8, а бабушка — за 10 минут. У них есть один фонарик. За какое наименьшее время вся семья сможет переправиться, если мост выдерживает только двоих? (Двигаться по мосту без фонарика, светить издали, а также носить друг друга на руках нельзя.)

Автор задачи — Константин Кноп, Санкт-Петербург

Кроссворд

Решите, пожалуйста, кроссворд.



По горизонтали

5. Первая буква греческого алфавита.
7. Системная (материнская)…
8. Единица измерения количества информации.
9. Второй экземпляр (файла и др.).
10. Структура данных — двусторонняя очередь.
12. Предписание, адресованное компилятору, а также приказ, указание, распоряжение.
15. Фигура вращения, похожая на бублик.
16. Совокупность данных на носителе, снабженная именем.
17. Конечное число точек на плоскости, соединенных отрезками кривых линий.
20. Особый ход в компьютерной игре, имитирующей древнеиндийскую игру.
23. Река в Сибири.
24. Буква латинского алфавита, которой обозначают одну из декартовых координат.
25. Число в системе условных обозначений символов.
28. Разработчик программы.
29. Порядковый номер байта оперативной памяти.
30. Так называли папку в операционной системе DOS, применявшейся в персональных компьютерах до разработки операционной системы Windows.

По вертикали

1. Вид связи.
2. Французская единица измерения расстояния на море, используемая в названии одного из романов Жюля Верна.
3. Величина изменения значения переменной цикла.
4. Взломщик компьютерных программ.
6. Пользователь телефонной линии.
7. Нестандартное устройство для вывода информации в компьютере.
11. Изображение, иллюстрирующее зависимость одной величины от другой.
12. Программа для обслуживания периферийного устройства.
13. Континент, на котором расположена страна, где находится фирма IBM.
14. Так называют информацию, которая передается по Интернету.
18. Электронная схема, применяемая в регистрах процессора для запоминания одного бита информации.
19. Утверждение, требующее доказательства.
21. В текстовом редакторе Microsoft Word — текст, набранный до нажатия клавиши Enter.
22. Определение, нахождение объекта по имени или другому признаку.
26. Структура данных, в которой применен принцип “последним пришел — первым вышел”.
27.

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

Числовой ребус “ИКС на ИКС”

Решите, пожалуйста, числовой ребус:

```
      И К С
    × И К С
    -------
      Р И С
    Д А Р
  И К С
  -------
  * * * * *
```

В нем, как принято в таких головоломках, одинаковыми буквами зашифрованы одинаковые цифры, разными буквами — разные цифры. Звездочкой (“*”) может быть любая цифра.

Указания по выполнению

1. Определите значение цифры **И** и возможные значения цифры **С**.

2. Исследуйте все возможные значения цифры **К**, рассмотрев произведение числа **ИКС** на **С**.

Ребусы в четверичной системе счисления

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

1.
```
  + C
    2
  D D
```

2.
```
  + X
    X
  Y 2
```

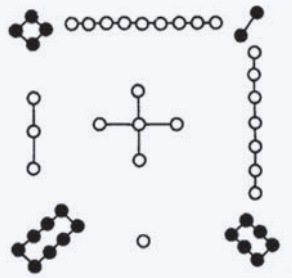
3.
```
  + 3
    B
  B E
```

4.
```
  + 3
    A
  B C
```

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

Странное изображение

Как вы можете прокомментировать изображение, представленное на рисунке ниже?



ПОИСК ИНФОРМАЦИИ

Три вопроса

Ответы на приведенные ниже вопросы найдите в Интернете или по другим источникам информации.

1. Какая кличка была у льва, снявшегося в фильме режиссера Эльдара Рязанова “Невероятные приключения итальянцев в России”?

2. О ком идет речь в русской пословице “Рядился… корове харчи поставлять, да за неустойку сам съел”?

3. Кому принадлежат слова: “Человеку не хватает мудрости остановиться на достигнутом”?

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

Переложить одну спичку[6]

Как, переложив одну спичку, получить верное равенство?

Семибуквенные слова

Вписав по горизонтали термины, соответствующие приведенным определениям, в закрашенном столбце вы получите слово, связанное с математикой и информатикой. Определите это слово и дайте ему определение.

1						
2						
3						
4						
5						
6						
7						

1. Устройство вывода информации в компьютере.

2. Утверждение, используемое в условных операторах, в функции ЕСЛИ и т.п.

3. Поиск и исправление ошибок в разрабатываемой программе.

4. Набор цветов, которые могут использоваться в изображении.

5. Элемент электронной таблицы.

6. Участок магнитного диска в виде двух концентрических окружностей, образуемый при разметке диска.

7. Государство в Европе.

Ответ присылайте в редакцию.

МЫСЛИ

Учение без размышления бесполезно, но и размышление без учения опасно.

Конфуций

Образование придает человеку достоинство…

Дени Дидро, французский философ

Не нужно доказывать, что образование — самое великое благо для человека. Без образования люди и грубы, и бедны, и несчастны.

Николай Гаврилович Чернышевский

Образованный человек тем и отличается от необразованного, что продолжает считать свое образование незаконченным.

Константин Михайлович Симонов

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

ЯПОНСКИЙ УГОЛОК

Два судоку

Решите, пожалуйста, две японские головоломки “судоку”:

1) простую:

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

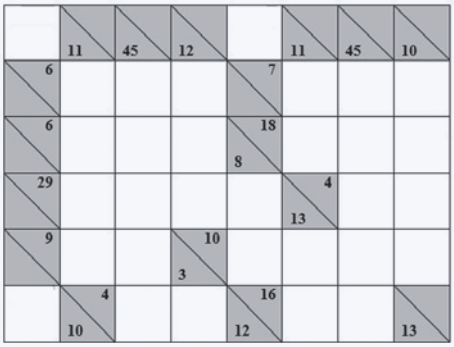
2) сложную:

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

Какуро

Какуро — также японская головоломка. Она представляет собой таблицу, аналогичную приведенной ниже. Оттененные клетки в какуро называются “легендой”. Они разделены наклонной чертой и содержат одно или два числа. Число в правом верхнем углу относится к прилегающему горизонтальному блоку клеток, а число в левом нижнем — к прилегающему вертикальному блоку.

Ее правила просты — надо вписать в пустые клетки цифры от 1 до 9 так, чтобы сумма в блоке соответствовала сумме в легенде. В блоке не могут стоять две одинаковые цифры. Так число 4 в легенде может состоять только из цифр 3 и 1, а не из цифр 2 и 2. Попробуйте решить приведенный какуро.



Решения (можно не всех головоломок) присылайте в редакцию.

ПРОЕКТНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА

Плотность записи информации на носителях ЭВМ

Александр Ушаков,
ученик гимназии № 1530 г. Москвы

Когда говорят о прогрессе электронных вычислительных машин (ЭВМ), то, как правило, подразумевают темпы роста производительности машин. Я же заинтересовался, как изменилась плотность записи на носителях информации.

Для этого я рассмотрел следующие носители:

1) перфоленты;
2) перфокарты;
3) магнитные ленты;
4) съемные магнитные диски ЭВМ серии ЕС;
5) гибкие магнитные диски (дискеты);
6) CD-диски;
7) DVD-диски.

Кратко опишу их.

1. Перфолента (перфорированная лента) — носитель информации в ЭВМ 1-го поколения в виде узкой бумажной ленты с отверстиями (*рис.* 1). Использовались также перфоленты на основе кино- и фотопленки.

Рис. 1

Отверстие на ленте соответствовало 1, его отсутствие — 0, то есть информация представлялась в двоичной системе счисления.

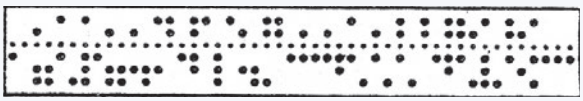


Рис. 2

Применялись 5-рядные (*рис.* 2), 6-рядные, 7-рядные и 8-рядные (см. *рис.* 1) перфоленты.

В середине ленты шла дорожка с более мелкой перфорацией — так называемая “транспортная дорожка”. Она служила для перемещения ленты с помощью зубчатого колеса.

Характеристики перфолент, рассчитанные мной на примере фрагмента 8-рядной ленты длиной 3,5 см, приведены в табл. 1:

Таблица 1

Тип	Площадь, см²	Емкость, байт	Плотность хранения информации, байт/см²
8-рядные	10,5	13	1,24

2. Перфокарта (перфорированная лента) — носитель в виде прямоугольника из тонкого картона размером 18,7 на 8,3 см. Она представляла информацию наличием или отсутствием отверстий в определенных позициях (см. *рис.* 3). Наибольшее распространение получили перфокарты с 12 рядами и 80 колонками. Также применялись перфокарты с 45 колонками.

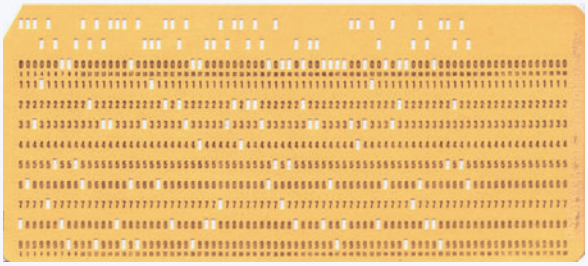


Рис. 3

При работе с перфокартами в двоичном режиме перфокарта рассматривалась как двумерный битовый массив; были допустимы любые комбинации пробивок.

При работе с перфокартами в текстовом режиме каждая колонка обозначала один символ; таким образом, одна перфокарта представляла строку из 80 символов. Допускались лишь некоторые комбинации пробивок. Для удобства работы с текстовыми данными вдоль верхнего края перфокарты часто печатались закодированные символы в "обычном" виде.

Характеристики перфокарт, полученные расчетным путем, приведены в табл. 2:

Таблица 2

Тип	Площадь, см²	Емкость, байт	Плотность хранения информации, байт/см²
Двоичный режим	157,25	120	0,76
Текстовый режим	157,25	80	0,51

3. Магнитная лента — носитель информации в виде гибкой ленты, покрытой тонким магнитным слоем (*рис.* 4). Информация на магнитной ленте фиксировалась посредством магнитной записи. Намагниченный участок ленты соответствовал единице, не намагниченный — нулю. Запись информации на ленту, ее считывание и стирание информации с ленты осуществлялись специальным устройством — магнитной головкой.

Впервые магнитная лента была использована для записи компьютерных данных в 1951 году фирмой Eckert-Mauchly Computer Corporation в ЭВМ UNIVAC I. В качестве носителя использовалась тонкая полоска металла шириной 12,65 мм, состоящая из никелированной бронзы. Плотность записи была 128 символов на дюйм.

Рис. 4

Существовали два основных типа лент — шириной 1 дюйм и 0,5 дюйма (1 дюйм ≈ 2,54 см).

Средняя плотность хранения информации на магнитных лентах составляла 62 байта/см².

4. В советских электронно-вычислительных машинах серии ЕС ЭВМ ("Единая серия ЭВМ")[7] и в их американских аналогах использовался съемный пакет магнитных дисков, в котором могло быть от 2 до 10 дисков (*рис.* 5–6).

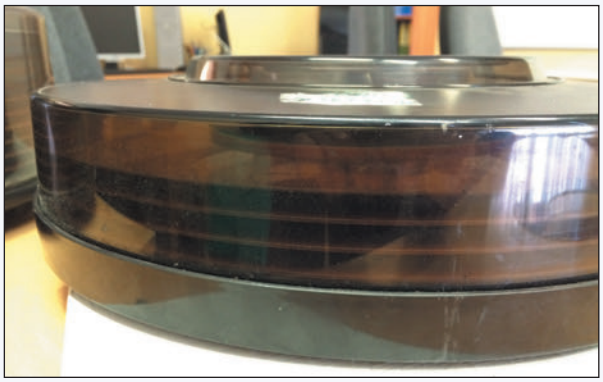


Рис. 5

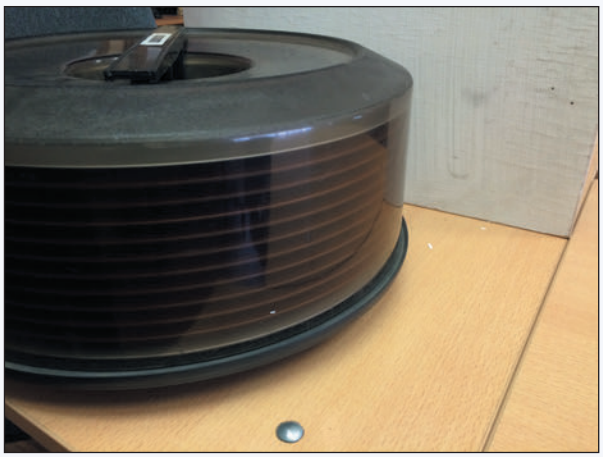


Рис. 6

Отдельный диск пакета (*рис.* 7) имел наружный размер 35,5 см, диаметр отверстия составлял 19 см.

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

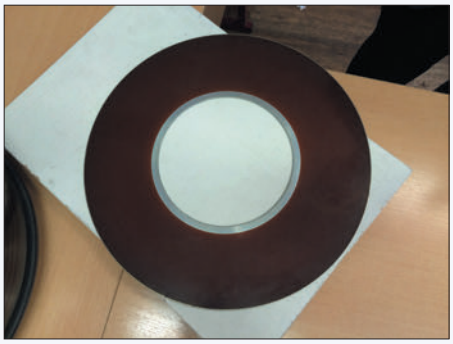


Рис. 7

Кстати, умельцы делали из этих дисков телевизионные антенны-“тарелки” (см. *рис*. 8).

Рис. 8

Средняя плотность хранения информации на дисках ЭВМ серии ЕС составляла 4456 байт/см^2.

5. Дискета (гибкий магнитный диск) представляла собой диск, покрытый ферромагнитным слоем и помещенный в защитный пластиковый корпус. Первая дискета диаметром 8 дюймов была представлена в 1971 году фирмой IBM. Затем были выпущены дискеты диаметром 5,25 и 3,5 дюйма (см. *рис*. 9).

Рис. 9

Характеристики дискет, полученные расчетным путем, приведены в табл. 3.

Средние значения плотности хранения информации для дискет трех указанных типов составляли, соответственно, 1760, 4600 и 24 200 байт/см^2.

6. Компакт-диск (англ. ***C**ompact **D**isc*, CD) — оптический носитель информации, разработанный и представленный в 1980 году. Он представляет собой поликарбонатную подложку толщиной 1,2 мм и диаметром 120 мм, покрытую тончайшим слоем металла (алюминий, золото, серебро и др.). Процесс записи и считывания информации осуществляется при помощи лазера. Данные кодируются и записываются в виде последовательности отражающих (“плоскость”) и не отражающих (впадина, которую называют “пит”) участков. Отражение интерпретируется как единица, впадина — как ноль. Эти впадины прожигает лазер в процессе записи информации (фрагмент поверхности CD-диска показан на *рис*. 10). Затем подложка металлизируется и покрывается слоем лака.

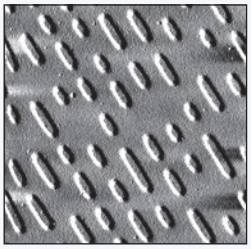


Рис. 10

Таблица 3

Диаметр, дюймов	Тип	Площадь, см^2	Емкость, байт	Плотность хранения информации, байт/см^2
8	обычной плотности	302	81 920	263
	обычной плотности		262 144	843
	обычной плотности		819 200	2634
	двойной плотности		1 024 000	3293
5,25	обычной плотности	126	112 640	846
	двойной плотности		368 640	2772
	четырехкратной плотности		737 280	5544
	высокой плотности		1 228 800	9239
3,5	обычной плотности	52	368 640	6456
	двойной плотности		737 280	12 912
	высокой плотности		1 474 560	25 824
	расширенной плотности		2 949 120	51 648

При чтении луч лазера, попадавший в пит, не отражается (0), а попавший на плоскость — отражается (1).

Характеристики CD-дисков приведены в табл. 4:

Таблица 4

Тип	Площадь, см²	Емкость, байт	Плотность, байт/см²
650 Мб	97	783 216 000	8 074 392
700 Мб	97	846 720 000	8 729 072
800 Мб	97	952 560 000	9 820 206
900 Мб	97	1 047 816 000	10 802 227

7. DVD-диск — носитель информации, выполненный в форме диска, имеющего такой же размер, как и компакт-диск, но более плотную структуру рабочей поверхности, что позволяет хранить больший объем информации. Это достигается за счет ряда факторов:

1) дорожки, на которых происходит запись, размещены более плотно — расстояние между ними уменьшено до 0,74 мкм, более чем в 2 раза по сравнению с 1,6 мкм для CD-дисков (см. *рис.* 11);

2) впадины (питы) также намного меньше: минимальная длина впадины — 0,4 мкм по сравнению с 0,834 мкм для CD;

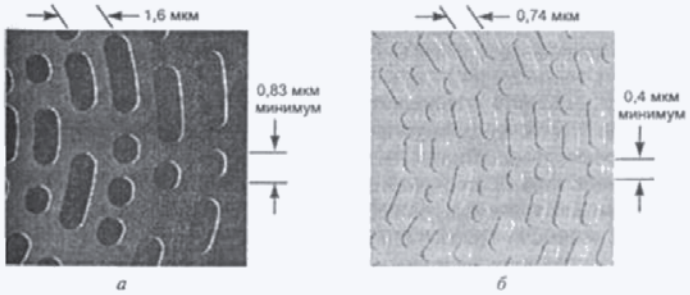


Рис. 11

3) DVD-диск позволяет считывать информацию более чем с одного слоя, изменяя фокусировку луча лазера чтения;

4) DVD позволяет использовать двусторонние диски.

Характеристики DVD-дисков приведены в табл. 5:

Таблица 5

Обозначение	Тип	Всего слоев	Площадь, см²	Емкость, Гб	Плотность хранения информации, байт/см²
DVD-1	SS SL	1	97	1,46	16 61 475
DVD-2	SS DL	1	97	2,66	29 444 879
DVD-3	DS SL	2	97	2,92	32 322 950
DVD-4	DS DL	2	97	5,32	58 889 758
DVD-5	SS SL	1	97	4,7	52 026 666
DVD-9	SS DL	1	97	8,54	94 533 559
DVD-10	DS SL	2	97	9,4	104 053 331
DVD-14	DS SL+DL	2	97	13,24	146 560 224
DVD-18	DS DL	2	97	17,08	189 067 117

Примечание. SS — односторонний (single-sided), DS — двухсторонний (double-sided), SL — однослойный (single-layer), DL — двухслойный (dual-layer).

Первые DVD-диски появились в ноябре 1996 года в Японии.

Сравнительный анализ полученных данных

На основе расчетов значений плотности хранения информации, проведенных для каждого типа носителя, определены средние значения, представленные в табл. 6.

Таблица 6

Носитель информации	Средняя плотность хранения информации, байт/см²
Перфоленты	1,4
Перфокарты	0,64
Магнитные ленты	315
Дискеты	10 200
Диски ЭВМ ЕС	4456
CD-диски	8 243 000
DVD-диски	80 340 000

Как показали расчеты, с момента выпуска первых электронно-вычислительных машин до настоящего времени плотность хранения информации на съемных носителях увеличилась более чем в 57 миллионов раз!

Использованные источники информации

1. *Антонов В.С., Соловьев С.П.* Электронная вычислительная машина ЕС-1050. М.: Статистика, 1976.
2. *Воройский Ф.С.* Информатика. Новый систематизированный толковый словарь-справочник. М.: Физматлит, 2003.
3. *Егоров Г.А., Песелев К.В.* СМ ЭВМ: комплексирование и применение. М.: Финансы и статистика, 1986.
4. *Смолевицкая М.Э.* Магнитные носители информации. / "Информатика", № 19–20/2008.
5. Интернет-ресурс *http://ru.wikipedia.org*.

ЗАДАЧНИК

Еще один "растительный" график

В одном из предыдущих выпусков "В мир информатики" был приведен график функции, по виду напоминающий листы щавеля. В данной статье рассмотрим еще один "растительный" график — график функции, которая в полярных координатах (см. [1]) имеет вид:

$$R = 3(1 + \cos^2(\varphi)) + 2\cos(\varphi) + \sin^2(\varphi) - 2\sin^2(3\varphi)\cos^4(\varphi/2).$$

Напомним, что в полярных координатах любая точка на плоскости определяется парой чисел: R — расстоянием до точки от полюса O (*рис.* 1) и φ — углом между полярной осью и прямой, соединяющей полюс и данную точку (угол φ измеряется в направлении против часовой стрелки от полярной оси).

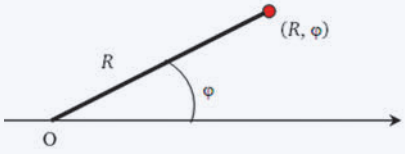


Рис. 1

Из уравнения в полярных координатах можно получить параметрические уравнения, то есть зависимость декартовых координат x и y от угла φ:

$$x = R\cos(\varphi) = (3(1 + \cos^2(\varphi)) + 2\cos(\varphi) + \sin^2(\varphi) - 2\sin^2(3\varphi)\cos^4(\varphi/2))\cos(\varphi);$$

$$y = R\sin(\varphi) = (3(1 + \cos^2(\varphi)) + 2\cos(\varphi) + \sin^2(\varphi) - 2\sin^2(3\varphi)\cos^4(\varphi/2))\sin(\varphi).$$

Используя параметрические уравнения линии, можно получить ее изображение с помощью программы Microsoft Excel или подобной. Фрагмент соответствующего листа имеет вид:

	A	B	C	D
1	**градусы**	**радианы**	**x**	**y**
2	0	0,00	8,00	0,00
3	5	0,09	7,82	0,68
4	10	0,17	7,33	1,29
…				
73	355	6,20	7,82	–0,68
74	360	6,28	8,00	0,00

Рис. 2

Примечания

1. Значения x и y рассчитываются согласно параметрическим уравнениям (см. выше).

2. В качестве аргумента функций Excel — *sin* (синус) и *cos* (косинус) — используется угол в радианах. Пересчет градусов в радианы должен проводиться с учетом того, что 360 градусов = 2π радиан.

По полученным расчетным данным можно построить график (тип — **Точечная с гладкими кривыми**).

Используя параметрические уравнения линии, можно также получить ее изображение в программе — для этого надо рассчитать значения координат x и y для всех углов от 1 до 360 градусов через 1 градус и поставить точку в соответствующем месте экрана.

Соответствующая программа на школьном алгоритмическом языке:

```
алг График_функции
нач цел x, y, угол, x0, y0, вещ угол2
|Координаты центра экрана
x0 := int(максX/2); y0 := int(максУ/2)
|Для всех целых значений углов
нц для угол от 1 до 360
  |Переводим угол в радианы
  угол2 := 6.28 * угол/360
  |Рассчитываем координаты
  |соответствующей точки кривой
  x := x0 + 3 * (1 + cos(φ)**2) +
       2 * cos(φ) + sin(φ)**2 -
       2 * sin(3φ)**2 * cos(φ/2)**4) *
       cos(φ)
  y := y0 - 3 * (1 + cos(φ)**2) +
       2 * cos(φ) + sin(φ)**2 -
       2 * sin(3φ)**2 * cos(φ/2)**4) *
       sin(φ)
  |и изображаем эту точку
  точка(x, y)
кц
кон
```

Примечания

1. *x*0 и *y*0 — координаты центра экрана (с учетом этих координат рассчитываются значения x и y); значения *x*0 и *y*0 зависят от величин *максX* и *максУ*, равных, соответственно, максимальному значению координат x и y в выбранном режиме работы экрана.

2. *угол*2 — величина угла в радианах.

3. Функция `int` возвращает целую часть ее вещественного аргумента.

4. Ось y на экране направлена вниз, поэтому в формуле для расчета координаты y использован знак "минус".

Задание для самостоятельной работы

Определите вид графика рассмотренной функции. Что он напоминает?

При использовании электронных таблиц следует учесть, что в них на диаграммах и графиках масштаб по осям x и y в общем случае не совпадает. Сделать его одинаковым можно, меняя размеры области диаграммы (пример для другой функции показан на *рис*. 3).

Полученный график и ответ на вопрос присылайте в редакцию. Фамилии всех приславших правильный ответ будут опубликованы.

Рис. 3

Литература

1. *Златопольский Д.М.* Полярные координаты. / “В мир информатики” № 193 (“Информатика” № 1/2014).

ЗАДАЧНИК

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

Джордж Пойя

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

Задача “Площадь квадрата”

Напомним условие: “В учебнике, изданном на планете β-Сириус, написано: «Площадь квадрата равна квадрату его стороны. Например, если сторона равна 11, то площадь равна 1001». Почему?”.

Решение

Ясно, что на планете используется недесятичная система счисления. Но какая? Обозначим ее основание — p.

Представим оба заданных в условии числа в развернутой форме записи:

$$11_p = p + 1;$$
$$1001_p = p^3 + 1.$$

Выражение $p^3 + 1$ — это сумма кубов двух чисел — p и 1. Поэтому можем записать:

$$p^3 + 1 = (p + 1) \times (p^2 - p + 1).$$

С другой стороны, согласно правилам расчета площадей на планете β-Сириус, площадь квадрата равна квадрату его стороны ☺. Поэтому $(p + 1) \times (p^2 - p + 1) = (p + 1)^2$, откуда можно получить:

$p^2 = 2p$, то есть $p = 2$.

Ответ: на планете β-Сириус применяется двоичная система счисления.

Можно также решить задачу подбором, подставив в развернутую форму записи чисел основание той или иной системы счисления.

Правильные ответы прислали:

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

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

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

— Гируцкий Павел, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Григоренко Дмитрий и Яснова Дарья, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

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

— Киселева Алена, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Красненкова Л.А.**;

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

Кроссворд (опубликованный в сентябрьском выпуске “В мир информатики”)

Ответы

По горизонтали: 1. Омега. 3. Текст. 7. Администратор. 8. Имя. 9. Шесть. 10. Фон. 13. Система. 14. Полиция. 17. Три. 18. Диаграмма. 19. Дек. 22. Катет. 23. Нолик. 24. Ять. 25. Икс. 26. Ось.

По вертикали: 2. Модем. 4. Слово. 5. Литера. 6. Отступ. 11. Регистр. 12. Элемент. 13. Строка. 15. Ячейка. 16. Шрифт. 20. Стек. 21. Плюс.

Ответы представили:

— Абрамов Алексей и Криволапов Тимофей, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

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

— Байбуза Дарья, Горелкина Лариса, Кузнецов Семен и Михайлова Алена, средняя школа села Ириновка, Новобурасский р-н Саратовской обл., учитель **Брунов А.С.**;

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

— Демидов Владимир и Радионов Александр, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Красненкова Л.А.**;

— Межогских Дарья, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Музычко Виктор, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Новожилов Игорь, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

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

— Храмцов Кирилл, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**

Задача "Банка с бактериями ☺"

Напомним условие: "В банке находятся бактерии. Через минуту каждая из бактерий делится пополам, затем каждая из получившихся бактерий через минуту делится пополам и т.д. Через час банка полна. Можно ли определить, через какое время банка была заполнена наполовину?".

Ответ: можно — это произойдет через 59 минут (каждую минуту число бактерий удваивается).

Задача "Еще одна банка с бактериями"

Напомним условие: "В банку поместили одну бактерию. Через минуту она поделилась пополам, затем каждая из получившихся бактерий через минуту делится пополам и т.д. Через час банка стала полной. Через какое время банка будет заполнена, если в нее поместить:

а) две бактерии?

б) четыре бактерии?".

Ответы:

а) через 59 минут;

б) через 58 минут.

Обоснование показано на рисунке ниже.

Ответы на задачи о бактериях представили:

— Азаров Даниил, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

— Васильева Мария, Павлов Степан и Яковлев Иван, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Деревянченко Дарья и Тихомирова Елизавета, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

— Евграфова Ксения, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Иванова Татьяна, Кривова Елена и Фурсова Алевтина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Красненков Артем, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Красненкова Л.А.**;

— Лаптева Ирина и Потанина Анна, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Ликина Людмила и Петриков Кирилл, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Марченко Елизавета и Федун Петр, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**

Обратим внимание, что не все присланные ответы — правильные.

Задача "Работники кафе"

Напомним, что следовало по ряду фактов определить профессию пяти работников кафе.

Ответ

Галкина — кассир, Шалкина — администратор, Малкин — официант, Палкин — повар, Балкин — кондитер.

Правильные ответы прислали:

— Акбарова Зульфия и Яковлев Иван, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

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

— Баранов Никита и Петриков Кирилл, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

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

— Бойкова Юлия, Ковалев Елисей, Попов Тимур, Сычева Ольга и Тютюнникова Диана, г. Воронеж, лицей № 2, учитель **Комбарова С.И.**;

— Великанов Андрей, Краснова Вера и Лопатников Дмитрий, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Деревянченко Дарья и Тихомирова Елизавета, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

— Егорова Александра, Костенко Петр и Чуркин Андрей, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

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

— Новиков Андрей и Цаплина Елена, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Чурикова Мария, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**

Задача "Постройка плотины"

Напомним, в задаче, предназначенной для учащихся 1–7-х классов, требовалось узнать, сколько друзей позвали три бобра на постройку плотины, если без друзей они построили плотину за 12 дней, а с ними — за 4 дня. Производительность всех бобров одинаковая.

Решение

Время строительства уменьшилось в три раза, значит, количество “строителей” увеличилось также в три раза. Значит, три бобра позвали еще шестерых.

Правильный ответ представили:

— Азаров Даниил и Криворучко Елена, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Васильева Мария, Павлов Степан и Яковлев Иван, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Евграфова Ксения и Краснова Вера, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Иванова Татьяна, Кривова Елена и Фурсова Алевтина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Красненков Артем, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Красненкова Л.А.**;

— Лаптева Ирина и Потанина Анна, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Ликина Людмила и Петриков Кирилл, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Марченко Елизавета и Федун Петр, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**

Задача “Кто на каком инструменте играет?”

Напомним условие: “Два мальчика умеют играть на гитаре, а еще один — на балалайке. На чем играет Андрей, если Максим и Павел играют на разных инструментах, а Павел и Андрей — тоже?”.

Решение

Опишем методику решения, приведенную в книге О.Б. Богомоловой “Логические задачи” (М.: БИНОМ. Лаборатория знаний, 2005), из которой заимствована задача.

Используем для решения такой эталон, который оформим на бумаге:

Дано:	**Рассуждения:**
Надо:	

Обозначим имена мальчиков символьными переменными А, П и М, а инструменты, на которых они играют, — б и г. Запишем эти данные в раздел “Дано:”. В задаче требуется узнать, на каком инструменте играет Андрей; запишем это в раздел “Надо:”.

Дано: Андрей (А) Максим (М) Павел (П) гитары (г) балалайка (б)	**Рассуждения:**
Надо: На чем играет Андрей?	

Рассуждения:

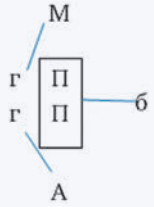
1) выделим в условии задачи повторяющуюся переменную. Это — имя Павел. Запишем две одинаковые соответствующие переменные друг под другом (см. эталон ниже);

2) так как по условию задачи Максим и Павел играют на разных инструментах, то над символьной переменной П запишем М. Аналогично, так как Павел и Андрей играют на разных инструментах, то под символьной переменной П запишем А;

3) выделим прямоугольником две одинаковые символьные переменные П;

4) запишем справа и слева переменные, которыми обозначены названия музыкальных инструментов;

5) так как по условию балалайка одна, значит, на ней играет Павел, следовательно, Максим и Андрей играют на гитарах. Тем самым мы нашли ответ на вопрос задачи — Андрей играет на гитаре.

Дано: Андрей (А) Максим (М) Павел (П) гитары (г) балалайка (б)	**Рассуждения:** 
Надо: На чем играет Андрей?	

Правильный ответ прислали:

— Азаров Даниил, Бирюков Сергей, Криворучко Елена и Таран Игорь, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Антонов Иван, Марченко Елизавета и Федун Петр, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

— Васильева Мария, Смолов Андрей и Яковлев Иван, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Евграфова Ксения и Краснова Вера, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Иванова Татьяна и Фурсова Алевтина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Красненков Александр, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Красненкова Л.А.**;

— Лаптева Ирина и Потанина Анна, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Ликина Людмила, Мазурова Светлана и Петриков Кирилл, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**

Новая задача. “Семья”

В семье трое детей — два мальчика и девочка. Их имена начинаются с букв “М”, “Н” и “О”. Имена, начинающиеся с букв “М” и “Н”, это имена одного мальчика и одной девочки. Имена, начинающиеся с букв “Н” и “О”, это имена одного мальчика и одной девочки. С какой буквы начинается имя девочки?

Задача “Есть ли такая цифра?”

Напомним, что следовало ответить на вопрос: “Есть ли такая цифра A, при которой $432_5 < 2A1_6$?”

Решение

Представим оба заданных числа в развернутой форме записи:

$$4 \times 25 + 3 \times 5 + 2 < 2 \times 36 + A \times 6 + 1,$$

откуда $107 < 73 + 6A$, или $34 < 6A$.

Так как A — цифра шестеричной системы счисления, то есть $A \le 5$, следовательно, последнее неравенство не может достигаться ни при каких значениях A.

Правильный ответ прислали:

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

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

— Гируцкий Павел и Тупицына Анна, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Григоренко Дмитрий и Яснова Дарья, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Зубов Владислав и Тимофеева Лидия, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

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

— Киселева Алена, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Красненкова Л.А.**;

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

Японские головоломки “судоку”, опубликованные в сентябрьском выпуске “В мир информатики”, решили:

— Абрамов Алексей и Бойко Николай, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

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

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

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

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

— Гируцкий Павел и Макарчук Сергей, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Деревянченко Дарья и Тихомирова Елизавета, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

— Киселева Алена, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Красненкова Л.А.**;

— Лопатин Михаил, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Межогских Дарья, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Новожилов Игорь, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Храмцов Кирилл, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**

Задача “Переправа”

Напомним условие.

На одном берегу реки собралась компания: отец с двумя сыновьями, мать с двумя дочерями и шериф с заключенным. Все они хотят переплыть на противоположный берег. При этом:

1) детишки не могут одни находиться на плоту;

2) шериф не может оставлять заключенного с остальными;

3) мужчина не может оставлять ни одного из двух сыновей одних с женщиной, а женщина — своих дочерей с мужчиной;

4) естественно, что плот не может плыть сам по себе, а на плоту могут находиться не более двух человек.

Какое минимальное количество раз плот причалит к противоположному берегу, чтобы перевезти всю компанию?

Ответ — 7.

Решение

1. Переправляются шериф и заключенный, шериф возвращается;

2. Переправляются шериф и отец, отец возвращается;

3. Переправляются отец и первый сын, отец возвращается;

4. Переправляются отец и второй сын, отец возвращается;

5. Переправляются отец и мать, мать возвращается;

6. Переправляются мать и первая дочь, мать возвращается;

7. Переправляются мать и вторая дочь.

Правильные ответы прислали:

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

— Лопатин Михаил, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Межогских Дарья, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**

Алексей, Дарья, Михаил будут награждены дипломами. Молодцы!

Кроссворд, опубликованный в майском выпуске “В мир информатики”, решила также Трус Анастасия, средняя школа села Дохновичи Брянской обл., учитель **Клопова Е.В.**

Программу, предложенную для самостоятельной работы в статье “Фрактал из окружностей”, опубликованной в августовском выпуске “В мир информатики”, представили:

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

— Комарова Мария и Новожилов Сергей, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

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

Комарова Мария из школы № 33 г. Ярославля разработала также программу, моделирующую решение головоломки “Ханойские башни” (статья с таким названием была опубликована в сентябрьском выпуске “В мир информатики”).

Учитывая сложность программ, редакция решила наградить Виктора, Владислава, Марию и Сергея дипломами.

Поздравляем всех награжденных и благодарим всех приславших ответы!

Списки приславших ответы на другие задания, опубликованные в сентябрьском выпуске “В мир информатики”, будут опубликованы в следующем выпуске журнала.

Школьные учителя

В школе работают учителя Волков, Сорокин и Коршунов. Каждый из них преподает два предмета, так что в расписании уроков у них всего шесть предметов: математика, физика, английский язык, литература, история и химия. Коршунов — самый молодой из них. Учитель химии старше учителя истории. Все трое — учитель химии, учитель физики и Сорокин — занимаются спортом. Когда учителя литературы и английского языка начинают обсуждать чемпионат России по футболу, то Коршунов тоже принимает участие в обсуждении. Сорокин не знает английского языка и не преподает математику. Кто какие предметы преподает?

Литература

1. *Богомолова О.Б.* Логические задачи. М.: БИНОМ. Лаборатория знаний, 2005.

Подарки Деда Мороза

Дед Мороз раздал детям 47 шоколадок так, что каждая девочка получила на одну шоколадку больше, чем каждый мальчик. Затем Дед Мороз раздал тем же детям 74 мармеладки так, что каждый мальчик получил на одну мармеладку больше, чем каждая девочка. Сколько всего было детей, если каждый получил больше одной сладости?

О цифрах

Какая цифра встречается чаще всего при записи первых ста натуральных чисел? А какая реже всего? Ответы получите, не выписывая все 100 чисел, а примените для расчетов электронную таблицу Microsoft Excel или подобную программу. Например, можно использовать такую таблицу:

	A	B	C	D	E		K	L
1			Цифра					
2	Десяток	Числа	0	1	2	…	8	9
3	1-й	1..10				…		
4	2-й	11..20				…		
5	3-й	21..30				…		
…	…	…	…	…	…	…	…	…
12	10-й	91..99				…		
13		100				…		
14		Всего				…		

Четыре сосуда с жидкостями

В чашке, стакане, кувшине и банке находятся молоко, лимонад, квас и вода. Известно, что вода и молоко не в чашке; сосуд с лимонадом стоит между кувшином и сосудом с квасом; в банке не лимонад и не вода; а стакан стоит между банкой и сосудом с молоком. Где что находится?

Лилии в пруду

Был пруд, в центре которого в один из дней вырос лист водяной лилии. Каждый день число таких листьев удваивалось, и на 30-й дней поверхность пруда уже была заполнена листьями лилий. Сколько дней понадобилось, чтобы заполнить листьями половину пруда? Сосчитайте также, сколько листьев выросло к десятому дню.

Друзья и города

Друзья *X*, *Y*, *Z*, *U*, *V* должны поехать в разные города — *А*, *Б*, *В*, *Г*, *Д*, *Е*. При этом *X* может ехать только в *А*, *Б*, *Д*; *Y* может ехать только в *Б* и *Г*; *Z* может ехать только один и в *В*; *U* не может ехать вместе с *Y*; *V* может ехать только с *X* и *Z*, но не в *Д*. В каком городе мог быть каждый из них, если оказалось, что вдвоем они не были ни в одном городе?

Покупка лошади

Некто продавал коня и просил за него 1000 рублей. Купец сказал, что за коня запрошена слишком большая цена. “Хорошо, — ответил продавец, — если ты говоришь, что конь дорого стоит, то возьми его себе даром, а заплати только за одни гвозди в его подковах. А гвоздей во всякой подкове по 6 штук. И будешь ты мне за них платить таким образом: за первый гвоздь полушку, за второй гвоздь заплатишь две полушки, за третий гвоздь — четыре полушки, и так далее за все гвозди: за каждый в два раза больше, чем за предыдущий”. Купец же, думая, что заплатит намного меньше, чем 1000 рублей, согласился.

Проторговался ли купец?

Примечание. Информацию о том, что такое “полушка”, найдите в Интернете или по другим источникам информации.

Блондины и брюнеты в классе

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

ШКОЛА ПРОГРАММИРОВАНИЯ

Решаем задачу “Необычный маршрут”

В прошлом учебном году в разделе “В мир информатики” была опубликована задача “Необычный маршрут”. Напомним, что в ней речь шла о перемещении мальчика Мити по такому маршруту по плоской равнине: 1 м — на юг, 2 м — на запад, 4 м — на север, 6 м — на восток, 7 м — на юг, 8 м — на запад, 10 м — на север, 12 м — на восток, 13 м — на юг, 14 м — на запад, 16 м — на север, 18 м — на восток и т.д. Требовалось установить:

1) на каком расстоянии от точки *A* будет Митя после 100 “шагов” (после прохождения 100 отрезков);

2) какова длина последнего, 100-го, отрезка пути;

3) какой общий путь пройдет Митя за 100 “шагов”.

Ответ на первый вопрос был обсужден в рубрике “Крепкий орешек” в ноябрьском выпуске журнала за 2014 год. Было показано, что можно установить, что за каждые четыре этапа маршрута мальчик удаляется от начала А на пять метров (использована теорема Пифагора). Следовательно, после 100 “шагов” Митя будет находиться на расстоянии 125 м от исходной точки.

Новая задача

Митя вышел из точки **A** плоской равнины и прошел 1 м — на юг, 2 м — на запад, 3 м — на север, 4 м — на восток, 5 м — на юг, 6 м — на запад, 7 м — на север, 8 м — на восток, 9 м — на юг, 10 м — на запад и т.д.

На каком расстоянии от точки **A** будет Митя после:

1) 100 “шагов” (после прохождения 100 отрезков);

2) 115 “шагов”;

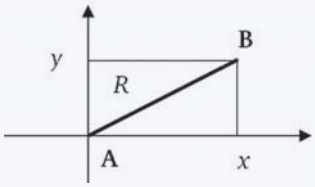
3) *n* “шагов”?

Получите ответы, используя электронную таблицу или разработав компьютерную программу (для ответа на 3-й вопрос — только программу).

Составим программу, с помощью которой можно определить, на каком расстоянии от исходной точки **A** будет находиться Митя после некоторого *n*-го “шага (после прохождения *n* отрезков).

Достаточно очевидное решение такое. Если ввести прямоугольную систему координат с началом в точке **A** и после каждого шага рассчитывать координаты нового местоположения мальчика, то при известных координатах *x* и *y* точки **B**, в которой будет находиться Митя после *n*-го шага, расстояние *R* от начальной точки **A** можно рассчитать, используя теорему Пифагора (см. рисунок ниже):

$$R = \sqrt{x^2 + y^2} \ .$$



С учетом этого схема решения задачи будет следующей:

1. Ввод значения *n*;
2. Для каждого шага от 1-го до *n*-го:

2.1. Расчет длины отрезка пути на очередном шаге (назовем эту величину — *длина_шага*);

2.2. Определение координат *x* и *y* места нахождения Мити после очередного шага.

3. Расчет значения *R*.
4. Вывод ответа.

На этапе 2.2 координаты *x* и *y* после очередного шага можно рассчитывать, зная соответствующие значения координат до этого шага:

1) при перемещении на юг:

```
y := y - длина_шага
```

2) при перемещении на запад:

```
x := x - длина_шага
```

3) при перемещении на север:

```
y := y + длина_шага
```

4) при перемещении на восток:

```
x := x + длина_шага
```

А как рассчитать длину отрезка пути, проходимого на том или ином шаге? На первый взгляд кажется, что никакой закономерности в последовательности значений отрезков нет. Однако это не так. Если объединить длины отрезков каждой, так сказать, “четверки шагов”, то можно составить таблицу:

Номер “четверки шагов”	**Номер шага в данной “четверке шагов”**			
	1	**2**	**3**	**4**
1	1	2	4	6
2	7	8	10	12
3	13	14	16	18
…	…	…	…	…

Из нее видно, что в каждой строке, начиная со второй, значения в том или ином столбце на шесть больше, чем в том же столбце предыдущей строки. Учитывая это, можем утверждать, что для четверки шагов с неким номером, который обозначим *номер*_4, длина отрезка может быть определена по формуле:

$$длина1 + 6 \times (номер\_4 - 1),$$

где — *длина*1 — значение, равное 1, 2, 4 или 6 в зависимости от номера шага в данной четверке шагов.

Читатели, знакомые с понятием “прогрессия”, конечно же увидели в последнем выражении формулу для расчета члена арифметической прогрессии с номером *номер*_4 при разности, равной 6.

В программе фрагмент, относящийся к расчету длины отрезка пути на очередном шаге, может быть оформлен с использованием оператора выбора (варианта):

```
выбор
. при номер = 1: длина_отрезка :=
                 1 + 6 * (номер_4 - 1)
. при номер = 2: длина_отрезка :=
                 2 + 6 * (номер_4 - 1)
. при номер = 3: длина_отрезка :=
                 4 + 6 * (номер_4 - 1)
. при номер = 4: длина_отрезка :=
                 6 + 6 * (номер_4 - 1)
все
```

— где *номер* — номер шага, который имеет в “своей” четверке шагов шаг с общим порядковым номером *номер_шага*.

Значение последней величины может быть определено на основе анализа такой таблицы:

Номер шага	Номер шага в данной “четверке шагов” (номер)
1	1
2	2
3	3
4	4
5	1
6	2
7	3
8	4
9	1
…	…

— из которой следует, что:

если остаток от деления значения *номер_шага* на 4 равен нулю
то
номер равен 4
иначе
номер равен указанному остатку
все

В программе на школьном алгоритмическом языке фрагмент для расчета значения *номер* имеет вид:

```
если mod(номер_шага, 4) = 0
  то
    номер := 4
  иначе
    номер := mod(номер_шага, 4)
все
```

— где `mod` — функция, возвращающая остаток от деления своего первого аргумента на второй. В других языках программирования для этого используется не функция, а специальная операция (как правило, знак этой операции также обозначается `mod`).

Менее очевидна формула для расчета номера четверки, в которую входит данный шаг. И здесь также составим таблицу для анализа:

Номер шага	Номер “четверки шагов” (номер_4)
1	1
2	1
3	1
4	1
5	2
6	2
7	2
8	2
9	3
…	…

Из нее следует, что

```
номер_4 := div(номер_шага + 3, 4)
```

или

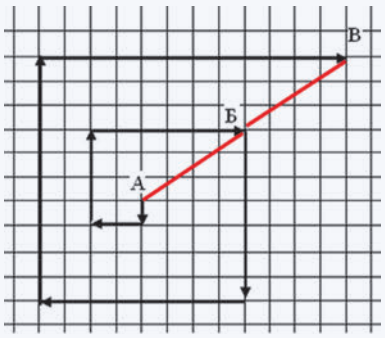
```
номер_4 := div(номер_шага - 1, 4) + 1
```

— где `div` — функция, определяющая целочисленное частное от деления своего первого аргумента на второй (в других языках программирования для этого также используется не функция, а специальная операция).

Предлагаем читателям самостоятельно убедиться в справедливости обоих вариантов.

Всю программу решения задачи (на языке программирования, который вы изучаете) соберите самостоятельно. С ее помощью определите, на каком расстоянии от исходной точки **А** будет находиться Митя после 50-го шага своего необычного маршрута.

Имеется также более рациональный (с точки зрения объема вычислений) вариант решения.



Если проанализировать первые четыре шага, то можно установить, что после них Митя будет находиться в точке **В**, от которой до точки **А** — 5 м (см.

формулу для расчета R выше). После восьми шагов (в точке **B**) мальчик будет находиться от точки **A** на расстоянии 10 м.

Учитывая это, может быть предложена такая идея решения задачи.

Если общее число шагов n кратно четырем, то искомое расстояние рассчитывается следующим образом:

$$R = 5 \times \text{div}(n, 4),$$

в противном случае нужно:

1) определить количество полных четверок шагов, сделанных Митей (оно равно div(n, 4));

2) найти координаты точки, в которой будет находиться мальчик после прохождения этого количества полных четверок шагов:

```
x := 4 * div(n, 4)
y := 3 * div(n, 4)
```

3) еще учесть 1, 2 или 3 шага так, как это делалось в первом варианте.

Новый вариант программы разработайте самостоятельно.

Результаты (можно не все) присылайте в редакцию.

ПРИ*1*Ы

* * *

Разговаривают в 60-х годах прошлого века два программиста:

— Как быстро возвести число в степень?

— Используя сдвиг!

— А если степень тройки?

— Очень просто! Покупаем вычислительную машину “Сетунь” или “Сетунь-70”...

(Указанные ЭВМ были основаны на троичной системе счисления.)

* * *

Сын спрашивает отца: “Сколько будет 1 + 1?”.

— Два.

— Неправильно — 10.

— Хорошо, согласен. Завтра куплю одну шоколадку и еще одну. Одну съем я, другую — мама, а остальные — ты.

* * *

Программист про двоичную систему счисления: “Да! В нашем алфавите 10 символов! И тут одно из 10 — либо вы его знаете, либо нет!”

* * *

В мире 10 категорий людей — те, которые понимают двоичную систему счисления, и те, которые ее не понимают.

Из переписки двух программистов

ВНИМАНИЕ! КОНКУРС!

Конкурс № 114

В качестве заданий этого конкурса предлагаем выполнить задания, предложенные для самостоятельной работы в статье “Еще один «растительный» график”.

Конкурс № 115

В качестве заданий этого конкурса предлагаем выполнить задания, предложенные для самостоятельной работы в статье “Решаем задачу «Необычный маршрут»” в рубрике “Школа программирования”.

* * *

Дорогие ребята! Присылайте ответы на вопросы наших конкурсов, даже если по каким-то причинам (поздняя доставка журнала или др.) вы подготовите их позже срока, указанного в условии конкурса. Мы обязательно рассмотрим все присланные ответы и при необходимости дополнительно отметим лучшие из них.

При оформлении ответов будьте, пожалуйста, внимательны — указывайте все необходимые сведения о себе (имя, пожалуйста, приводите полностью), населенном пункте, учебном заведении и учителе информатики.

Ответы отправьте в редакцию до 15 февраля по адресу: 121165, Москва, ул. Киевская, д. 24, “Первое сентября”, “Информатика” или по электронной почте: *vmi@1september.ru*.

Общероссийский проект

Школа цифрового века

Министерство образования Московской области ● Издательский дом «ПЕРВОЕ СЕНТЯБРЯ»

Бесплатные электронные учебники — каждому ученику на 2016 год!

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.

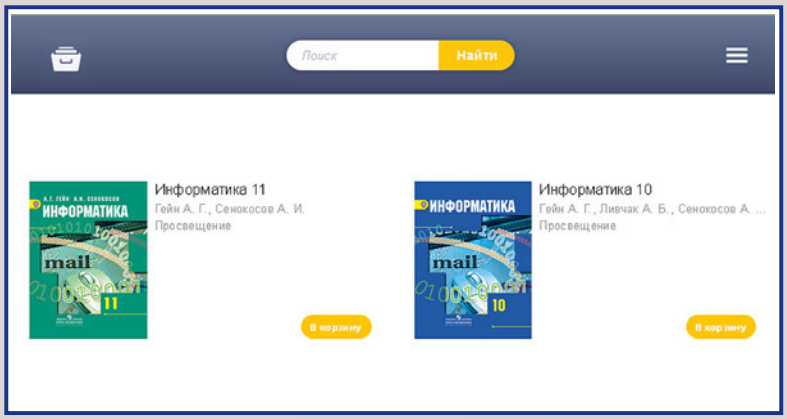
Следите за проектом!

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»