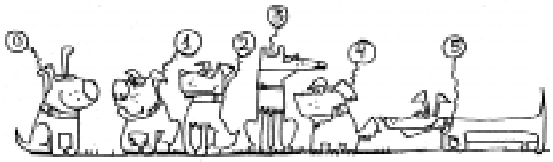
Паньгина Нина Николаевна
Паньгин Андрей Александрович

ЗАНЯТИЕ 3. РАЗРАБОТКА ПРИЛОЖЕНИЙ НА VISUAL BASIC (ОБУЧАЮЩИЕ И ТЕСТИРУЮЩИЕ ПРОГРАММЫ)

Вы хотите как можно быстрее научиться создавать программные приложения под Windows? Поверьте в свои силы и создайте первые «полезные» программы, то есть такие программы, которые могут использовать школьные учителя на своих уроках для обучения и проверки знаний учащихся.

ИСПОЛЬЗОВАНИЕ МАССИВА ОБЪЕКТОВ И ГЛОБАЛЬНЫХ ПЕРЕМЕННЫХ



Кроме обычных массивов, хранящих данные различного типа, в VB разрешается определять массивы объектов (control arrays) (элементов управления), что весьма удобно, если в программе имеются группы объектов, действующих примерно одинаково. Такие массивы позволяют «привязывать» разные элементы управления к одной процедуре обработки события. Например, если в программе создан массив из нескольких командных кнопок, щелчок на любой из этих объектов вызывает одну и ту же процедуру обработки события Click. В то же время VB дает возможность различать конкретные объекты в массиве – это достигается передачей в процедуру индекса нужного элемента.

Каким же образом создать массив объектов? Проще всего это сделать так:

- нанести объект на форму;
- скопировать его в буфер памяти (пункт Edit горизонтального меню, затем Copy);
- произвести вставку объекта из буфера памяти (Edit, Paste);
- при появлении диалогового окна с вопросом «У Вас уже есть объект с таким именем. Вы желаете создать массив объектов?» ответить Да (Yes).

В созданном массиве VB присвоил каждому объекту индекс по порядку. Это отразилось в значениях свойства Index у каждого объекта в окне Properties. Нумерация начинается с нуля. Чтобы обратиться к элементу массива объектов, необходимо указать имя массива, а за ним – индекс в круглых скобках, например:

```
Command1(0).Caption = "Кнопка 0",
Command1(1).Caption = "Кнопка 1".
```

Создание массива объектов можно упростить. Для этого следует вычертить первый элемент управления и полностью описать его свойства, затем скопировать и расставить на форме уже готовые копии (с нужными свойствами и размерами). Создание массива объектов программным путем будет рассмотрено позднее.

Задание 1. Тест по русскому языку.

Создать приложение, являющееся тестом по русскому языку на слова с безударными гласными в корне слова. Слова с пропущенными буквами должны выводиться в текстовом окне по очереди при каждом нажатии на кнопку «Введите слово». Тестируемый должен выбрать нужную букву-кнопку (А, О, Е, И) и нажать на нее. В окне метки слева внизу должен появляться комментарий в виде слов (Верно/Неверно). По окончании теста в текстовом окне выводится оценка с указанием количества правильных ответов. Программа завершает работу по нажатию на кнопку «Выход».

Выполнение задания 1.

Первая часть – *визуальное программирование*.

1. Необходимо создать форму и переименовать ее в соответствии с рисунком 1, свойству Caption (название) присвоив значение «Русский язык».

2. Нанести на форму Метку 1, изменив ее свойство Caption на строку «Вставьте пропущенную букву», свойству Alignment (выравнивание) придать значение Center (по центру), изменить значения свойства Font (шрифт), выбрав название шрифта Times New Roman, размер 16, стиль полужирный курсив.

3. Нанести на форму Метку 2, очистить ее (свойству Caption присвоить пустую строку), изменить значения свойства Font по своему усмотрению.

4. Нанести на форму Текстовое окно, очистить его, свойству Alignment придать значение Center, не забыв присвоить свойству Multiline значение True, изменить значения свойства Font, выбрав название шрифта Times New Roman, размер 16, стиль полужирный.

5. Нанести на форму массив из четырех Командных кнопок 1, изменив их размеры и свойство Font, а также свойство Caption на «А», «О», «Е», «И».

6. Нанести на форму Командные кнопки 2 и 3, изменив их размеры и свойство Font, а также свойство Caption на «Введите слово» и «Выход» соответственно.

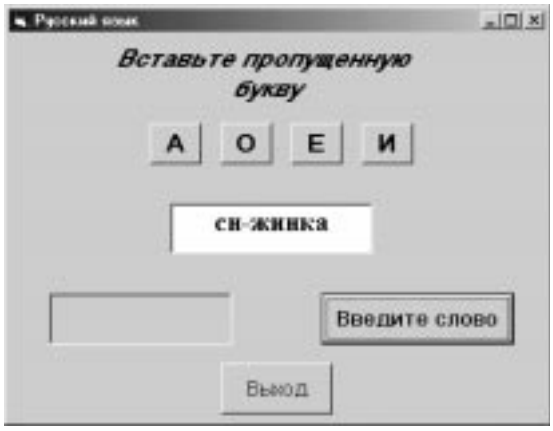


Рисунок 1

Вторая часть – *написание кода программы.*

7. В данном приложении будут использоваться два глобальных массива: массив из 10 слов с пропущенными буквами SL$(10) и массив B(10), содержащий номера пропущенных букв (равные индексам элементов массива кнопок 1). В приложении также будут использоваться глобальные переменные: N – номер очередного слова, K – количество правильных ответов. Описываем все глобальные переменные в общем разделе описаний General Declarations:

```
Dim SL$(10), B(10), K, N
```

8. При событии загрузки формы выполняется инициализация переменных и объектов. Необходимо присвоить (или ввести из файла) значения элементам обоих массивов. Переменным N и K можно присвоить нулевые начальные значения.

```
Private Sub Form_Load ()
  N = 0: K = 0
  SL$(1) = "г-лова": B(1) = 1
  SL$(2) = "св-детель": B(2) = 3
  SL$(3) = "в-тражи": B(3) = 3
  SL$(4) = "ед-ница": B(4) = 3
  SL$(5) = "тр-ва": B(5) = 0
  SL$(6) = "сн-жинка": B(6) = 2
  SL$(7) = "адв-кат": B(7) = 1
  SL$(8) = "диск-тека": B(8) = 1
  SL$(9) = "м-шина": B(9) = 0
  SL$(10) = "св-тильник": B(10) = 2
End Sub
```

9. В процедуре обработки события Щелчок на кнопку «Введите слово» должен увеличиться на единицу номер очередного слова, а само слово должно появиться в текстовом окне (свойству text Текстового окна необходимо присвоить значение элемента массива SL с номером N). В случае, если номер слова превысил допустимое значение (в данном случае 10), необходимо вывести сообщение о количестве правильных ответов. Не забыть перед выводом очередного слова в Текстовое окно очистить его и Метку 2, предназначенную для комментариев.

```
Private Sub Command2_Click()
   Text1.Text = ""
   Label2.Caption = ""
   N = N + 1
   If N > 10 Then Label2.Caption =_
   "Из 10 вопросов верных ответов "_
   + Str$(K): Exit Sub
   Text1.Text = SL$(N)
End Sub
```

Примечание. Поскольку ширина страницы книги не бесконечна, некоторые операторы программы разбиты на несколько строк. В конце таких строк используется символ подчеркивания (_), который является стандартным символом продолжения языка Visual Basic. При вводе текста в окно редактора вы вполне можете опустить символы продолжения и записать оператор в одной строке. Обе записи правомерны, и компилятор правильно их воспримет.

10. Процедура обработки события Щелчок на кнопку-букву является одной для всего массива, информация о конкретном элементе массива содержится в параметре Index. В процедуре обработки данного события необходимо проверить, верная ли буква-кнопка нажата, а эта проверка заключается в том, чтобы сравнить значение индекса нажатой кнопки со значением элемента массива B с номером очередного слова. В случае верного ответа в метку занести слово «ВЕРНО» и увеличить на единицу количество правильных ответов, в случае неверного ответа – в метку занести слово «НЕВЕРНО».

```
Private Sub Command1_Click (Index
As Integer)
   If Index = B(N) Then
   Label2.Caption = "ВЕРНО"
   K = K + 1
   Else
   Label2.Caption = "НЕВЕРНО"
   End If
End Sub
```

11. В процедуре обработки события Щелчок на кнопку «Выход» необходимо завершить работу программы.

```
Private Sub Command3_Click()
   END
End Sub
```

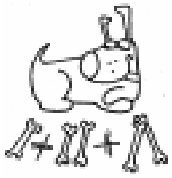
12. Для того чтобы после неверного выбора буквы нельзя было еще раз нажимать кнопку, можно внести дополнение в код процедуры Command1_Click, сделав буквы-кнопки недоступными.

```
For i = 0 To 3
   Command1(i).Enabled = False
Next I
```

13. Снова сделать их доступными (свойству Enabled присвоить значение True) нужно будет при вводе нового слова в процедуре Command2_Click.

14. Запустить программу на выполнение, протестировать ее и завершить.

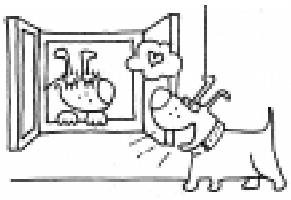
Задание 2. Тест по арифметике (для самостоятельного выполнения).



Создать приложение, согласно рисунку 2, представляющее собой тест по арифметике на действия сложения и вычитания с целыми числами в пределах от 0 до 99. Числа и знак операции формируются случайным образом. Примеры должны выводиться в текстовое окно по очереди при каждом нажатии на кнопку «Следующий пример». Ответ в текстовое окно можно набирать либо на клавиатуре, либо с помощью мышки, нажимая нужную цифру-кнопку и, если необходимо, знак «минус». Убедившись в правильности набора, нажать кнопку «Ввод ответа». В окне метки слева внизу должен появляться комментарий в виде слов (Правильно/Неправильно). По окончании теста в этом же окне метки появляется оценка. Программа завершает работу по нажатию на кнопку «Выход».

Рисунок 2

ФУНКЦИЯ И ОКНО INPUTBOX



Текстовое поле требуется для ввода разнообразной информации. Но иногда требуется ввести только краткую информацию, например, имя пользователя или значения даты и времени. Для ввода небольших фрагментов текста VB предлагает функцию вызова Окна ввода ***InputBox***.

Окно InputBox состоит из четырех элементов (рисунок 3):

- строка заголовка;
- приглашение к вводу;
- поле ввода со значением, предлагаемым по умолчанию;
- две кнопки (OK и Cancel).

Функция вызова окна InputBox имеет следующий синтаксис с соответствующими именованными аргументами:

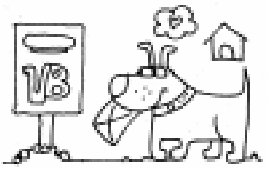
```
Возвращаемое значение = InputBox_
(prompt [,title] [,default] [,xpos]_
[,ypos])
```

Параметр prompt определяет текст, отображающийся в диалоговом окне как приглашение. Title отвечает за надпись заголовка; если этот параметр не указан, то отображается название приложения. Параметр default определяет значение по умолчанию, отображаемое в строке ввода. Параметры xpos и ypos указывают координаты верхнего левого угла окна (по умолчанию окно появляется посредине экрана). Параметры xpos и ypos нужно использовать совместно. Например:

```
strReturn = InputBox ("Введите_
десятичное число", "Окно ввода",_
"12345")
```

Функция InputBox возвращает строку, введенную пользователем.

ОКНО MESSAGEBOX



Для вывода различных сообщений имеется окно, подобное InputBox, – ***MessageBox***. Почти все приложения Windows используют MessageBox, так как этот компонент входит в состав Windows, а VB только предоставляет возможность его вызова.

Вид окна MessageBox может быть различным (рисунок 4), но в его состав всегда входят:

- текст сообщения;
- заголовок;
- пиктограмма (может отсутствовать);
- набор кнопок.

MessageBox можно вызывать как процедуру (или оператор) и как функцию.

Синтаксис процедуры:

```
MsgBox Prompt [, Buttons] [, Title]
```

Например:

```
MsgBox "Десятичное число 12345 =" +_
Chr(13) + Chr(10) + "двоичному числу_
= 11000000111001", 1, "Окно вывода"
```

Синтаксис функции:

```
Возвращаемое_значение = MsgBox_
(Prompt [,Buttons] [,Title])
```

По определению, функция возвращает некоторое значение, а оператор –

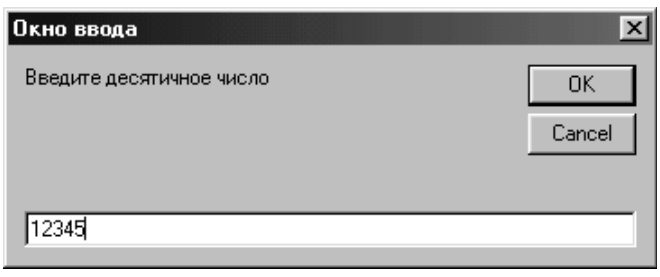


Рисунок 3

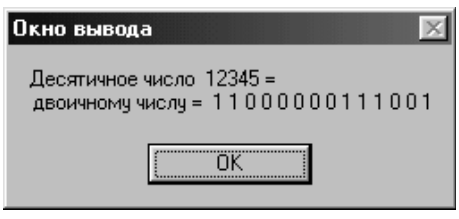


Рисунок 4

нет. Поэтому для запросов у пользователей некоторой информации требуется вызывать MsgBox как функцию, так как возвращаемое значение используется для определения кнопки, нажатой пользователем, например:

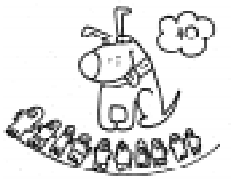
```
Select Case MsgBox ("Вопрос",_
vbAbortRetryIgnore + vbQuestion,_
"Заглавие")
  Case vbAbort
    Call Abort
  Case vbRetry
    Call Retry
  Case vblgnore
    Call Ignore
End Select
```

Значения, возвращаемые функцией MsgBox:

Константа	*Значение*	*Нажата кнопка*
vbOK	1	OK
vbCancel	2	Отмена
vbAbort	3	Стоп
vbRetry	4	Повторить
vblgnore	5	Пропустить
vbYes	6	Да
vbNo	7	Нет

Окно MsgBox всегда требует реакции пользователя на отображенное сообщение, поэтому надо помнить, что, пока на экране находится Окно сообщения, программа переходит в состояние ожидания.

Задание 3. Перевод чисел из десятичной системы счисления в двоичную.



Создать приложение, представляющее собой программу по переводу чисел из десятичной системы счисления в двоичную. На форме расположить две командные кнопки с названиями «Перевод» и «Выход». При каждом нажатии на кнопку «Перевод», появляется Окно ввода, как на рисунке 3. Пользователь должен набрать десятичное число и нажать на кнопку OK. После этого должно появиться Окно вывода, как на рисунке 4, с числом в двоичном виде. Программа завершает работу по нажатию на кнопку «Выход».

Выполнение задания 3.

Первая часть – *визуальное программирование*.

1. Необходимо создать форму и свойству Caption присвоить значение «Перевод чисел из десятичной системы счисления в двоичную».

2. Нанести на форму Командные кнопки 1 и 2, изменив их размеры и свойство Font, а также свойство Caption на строки «Перевод» и «Выход», соответственно.

Вторая часть – *написание кода программы.*

3. В процедуре обработки события Щелчок на кнопку «Перевод» необходимо вызвать функцию InputBox, присвоить возвращаемое значение числовой переменной n и запомнить его в строковой переменной n1$. Перевод осуществить в цикле Do Loop Until, находя остатки от деления на 2 и накапливая их в строковой переменной n2$ справа налево. Вывод можно осуществить в Окно вывода с помощью оператора MsgBox. Для перехода на другую строку использовать символы перевода каретки и конца строки Chr(13) и Chr(10) или константу vbCrLf.

```
Private Sub Command1_Click()
  n = InputBox ("Введите десятичное_
  число ", "Окно ввода")
  If n = "" Then Exit Sub
  n1$ = Str(n)
  Do
    ost = n Mod 2
    n = n \ 2
    n2$ = Str(ost) + n2$
  Loop Until n = 0
  MsgBox "Десятичное число " + n1$_
  + " =" + Chr(13) + Chr(10) + _
  "двоичному числу = " + n2$,_
  vbDefaultButton1, "Окно вывода"
End Sub
```

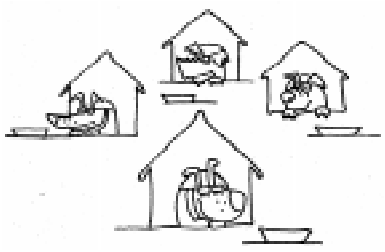
4. В процедуре обработки события Щелчок на кнопку «Выход» необходимо

завершить программу оператором END.

```
Private Sub Command2_Click()
   End
End Sub
```

5. Запустить программу на выполнение, протестировать ее и завершить.

РАЗРАБОТКА ПРИЛОЖЕНИЙ С НЕСКОЛЬКИМИ ФОРМАМИ



Как правило, типичное приложение имеет более одной формы. Когда оно начинает работу, загружается главная форма. После загрузки форма захватывает требуемые ресурсы, поэтому неиспользуемые формы из памяти можно выгружать. Но если формы содержат, например, большие растровые изображения, которые загружаются и выгружаются довольно долго, то имеет смысл такие формы держать в памяти и отображать по мере надобности.

Для загрузки и выгрузки формы используются операторы Load и Unload.

Синтаксис операторов:

```
Load FormName
Unload FormName
```

Для отображения и скрытия формы используются методы Show и Hide.

```
FormName.Show
FormName.Hide
```

Задание 4. Обучающая программа по русскому языку.

Создать приложение, являющееся обучающей программой по русскому языку по теме «Деепричастие и деепричастный оборот». Имеется следующий текст, который необходимо представить в качестве учебного материала:

1. «Деепричастие – это особая форма глагола, обозначающая добавочное действие основного глагола».

2. «Деепричастный оборот – это деепричастие вместе с зависимыми словами».

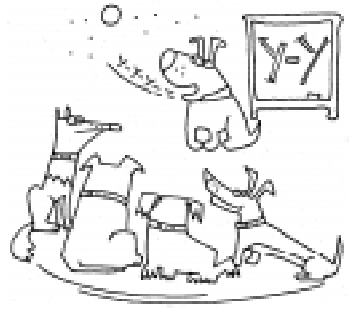
3. «Деепричастный оборот выделяется двумя запятыми, когда он находится в середине предложения, и одной запятой – в начале и в конце предложения».

4. «Не выделяется запятыми одиночное деепричастие, если оно стоит в конце предложения сразу после основного глагола».

5. «Не выделяется запятыми деепричастный оборот, если он является фразеологическим оборотом».

Для того чтобы приступить к написанию обучающей программы, необходимо следующее:

- разработать художественное оформление, то есть создать ряд рисунков, сопровождающих объяснения, расположить текст и рисунки на одной или нескольких формах, используя различные объекты управления и контроля;
- разработать сценарий, то есть последовательность действий, которые должен выполнять обучаемый пользователь, следуя указаниям-подсказкам программы, и последовательность событий, которые должны происходить в результате различных действий пользователя.



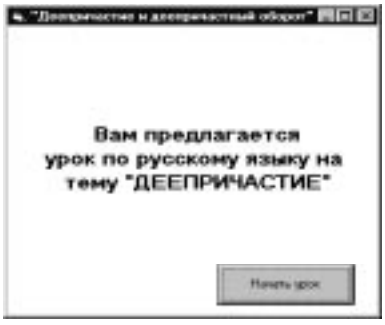


Рисунок 5

Рисунок 6

Рисунок 7

Рисунок 8

Рисунок 9

Начало программы может выглядеть, например, как на рисунке 5. Страница, содержащая определение деепричастия, может выглядеть, как на рисунке 6.

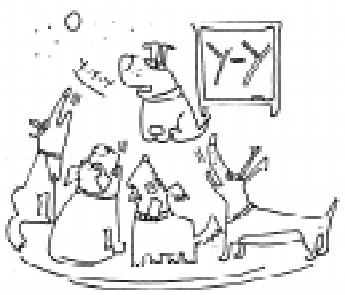
Для того чтобы лучше объяснить деепричастный оборот, можно создать два поясняющих рисунка. На одном изобразить машину (деепричастие), а на втором – прицеп (зависимые слова), применить анимационный эффект (машина подъезжает, зацепляет прицеп и везет его). Это будет смотреться, как на рисунке 7.

Далее следует привести правила по выделению запятыми деепричастных оборотов. Для этого нужно подобрать соответствующие примеры, сопроводительные рисунки, применить анимационные эффекты. В результате форма может принимать вид, как на рисунках 8–11.

Выполнение задания 4.

Первая часть **–** *визуальное программирование*.

При рассмотрении рисунков 5–11 можно заключить, что удобнее было бы создать не одну, как это делалось раньше, а две формы. Первая форма должна отражать начало программы (рис.5, 6), а вторая форма – основную часть (рис.7–11).



1. Переименовать первую форму в соответствии с рисунком 5, свойству Caption присвоив значение «Деепричастие и деепричастный оборот».

2. Нанести на форму Метки 1 и 2, Метку 1 очистить, а в Метку 2 поместить строку «Вам предлагается урок по русскому языку на тему ДЕЕПРИЧАСТИЕ». Свойству Alignment обеих меток придать значение Center, изменить значения свойства Font, выбрав название шрифта MS Serif, стиль полужирный, размер 18 и 14 для Меток 1 и 2, соответственно.

3. Нанести на форму Командную кнопку, изменив ее свойство Caption на строку «Начать урок».

4. Через пункт меню Add Form (Добавить Форму) добавить новую форму New Form (по умолчанию свойство Name этой формы – Form2), изменить свойство Caption, как в первой форме.

5. Нанести на вторую форму Метку, занести в нее текст «Деепричастный оборот», свойству Alignment придать значение Center, свойству Autosize – True, изменить значения свойства Font, выбрав название шрифта MS Serif, размер 16, стиль полужирный.

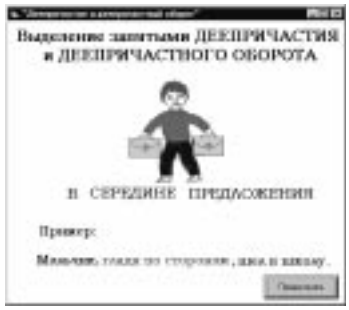


Рисунок 10

Рисунок 11

6. Нанести на форму три Окна рисунка 1, 2 и 3, придав свойству Autosize значение True.

7. Нанести на форму Командную кнопку, изменив ее свойство Caption на «Продолжить».

8. Для осуществления эффектов анимации (самая простая имитация движения – периодическая смена двух кадров) необходимо нанести на форму три таймера, по срабатыванию которых и будут меняться изображения.

Вторая часть – *написание кода программы.*

9. Алгоритм данной программы строится согласно последовательности действий пользователя, который, по мере обучения, щелкает на Командные кнопки с названиями «Начать урок», «Продолжить» и т.д. В зависимости от порядкового номера щелчка, должен меняться внешний вид формы, либо должна загрузиться другая форма. Глобальную переменную, известную для всех форм, необходимо объявлять как Public или Global в модуле (отдельном файле с расширением .bas). Для этого в пункте меню Add Module (Добавить модуль) открыть новый модуль New Module и записать в общем разделе объявлений модуля следующую строку:

```
Public k
```

10. В процедуре обработки события Щелчок на Командную кнопку «Начать урок» на первой форме должен увеличиться на единицу номер щелчка (переменная k). В зависимости от значения k выполняются различные действия.

- При k=1 свойству Caption Метки 1 присвоить слово «ОПРЕДЕЛЕНИЕ», а в Метку 2 занести само определение деепричастия, обвести его в рамку путем замены свойства BorderStyle, поменять название Командной кнопки на «Продолжить».
- При k=2 необходимо передать управление на Форму 2, для чего сначала выгрузить Форму 1, а затем показать Форму 2 (загрузка формы при методе Show происходит автоматически). Во второе и третье Окна рисунков загрузить картинки машины и прицепа, расположив их по левую и правую стороны формы. Для осуществления движения машины по направлению к прицепу включить, то есть сделать доступным, Таймер 1, по которому и осуществляется именно это движение.

```
Private Sub Command1_Click()
  k = k + 1
  Select Case k
  Case 1
    Label1.Caption = "ОПРЕДЕЛЕНИЕ"
    Label2.Caption = "Деепричастие_
    – это особая форма глагола,_
    обозначающая добавочное_
    действие основного глагола"
    Command1.Caption = "Продолжить"
    Label2.BorderStyle = 1
  Case 2
    Unload Form1
    Form2.Show
    Form2.Picture2.Picture  =_
    LoadPicture("deepr1.bmp")
    Form2.Picture3.Picture  =_
    LoadPicture("deepr2.bmp")
    Form2.Picture2.Left   =_
    -Form2.Picture2.Width
    Form2.Picture3.Left=_
    Form2.Width- Form2.Picture3.Width
    Form2.Timer1.Enabled = True
  End Select
End Sub
```

11. После выгрузки первой формы происходит показ второй формы. В общем разделе описаний для второй формы необходимо описать константу h – шаг для перемещения рисунков по экрану – и глобальную переменную n, используемую для смены рисунков при изображении движения.

```
Dim n
Const h = 50
```

12. В процедуре обработки события Timer для Таймера 1 должно перемещаться Окно рисунка 2, содержащее изображение машины. Это удобно делать, изменяя свойство Left на величину шага h до тех пор, пока Окно рисунка 2 (машина) не приблизится к Окну рисунка 3 (прицеп).

```
Private Sub Timer1_Timer()
  If Picture2.Left < Picture3.Left_
  - Picture2.Width Then
     Picture2.Left = Picture2.Left + h
  End If
End Sub
```

13. В процедуре обработки события Щелчок на кнопку «Продолжить» второй формы увеличивается на единицу значение переменной k, которое затем анализируется.

- При k = 3 Окно рисунка 2 соединяется с Окном рисунка 3, выключается Таймер 1 и включается Таймер 2, по которому осуществляется совместное движение Окон рисунков (машины с прицепом). В Окно рисунка 1 загружается картинка с текстом-примером деепричастного оборота, Окно рисунка 1 становится видимым (первоначально свойству Visible было присвоено значение False).
- При k = 4 останавливается Таймер 2, в Метку заносится текст «Выделение запятыми ДЕЕПРИЧАСТИЯ и ДЕЕПРИЧАСТНОГО ОБОРОТА», Окно рисунка 3 становится невидимым. В Окно рисунка 1 загружается картинка-пример с деепричастным оборотом в начале предложения, в Окно рисунка 2 загружается изображение мальчика с портфелем и включается Таймер 3, отвечающий за перемещение по Форме Окна рисунка 2.
- При k = 5, 6, 7 в Окно рисунка 1 загружаются последующие картинки-примеры, а Окно рисунка 2 занимает начальное положение справа от Формы (Таймер 3 продолжает работать).
- При k > 7 появляется Окно сообщения MSGBOX с вопросом «Вы хотите закончить обучение?» и предлагается два варианта ответа: Да (Yes) и Нет (No). При выборе первого варианта обучающая программа завершает свою работу, а при выборе второго варианта программа начинает свою работу с самого начала, для чего необходимо обнулить переменную k, выгрузить вторую форму и показать первую.

```
Private Sub Command1_Click()
  k = k + 1
  Select Case k
  Case 3
    Picture2.Left = Picture3.Left_
    - Picture2.Width
    Timer1.Enabled = False
    Timer2.Enabled = True
    Form2.Picture1.Picture =_
    LoadPicture("deepr3.bmp")
    Form2.Picture1.Visible = True
  Case 4
    Timer2.Enabled = False
    Form2.Label1.Caption =_
    "Выделение запятыми _
    ДЕЕПРИЧАСТИЯ" + Chr(13) + _
    " и ДЕЕПРИЧАСТНОГО ОБОРОТА"
    Form2.Picture1.Picture =_
    LoadPicture("deepr4.bmp")
    Form2.Picture2.Picture =_
    LoadPicture("Ris4-0.bmp")
    Form2.Picture3.Visible = False
    Form2.Picture2.Left =_
    Form2.Width
    Form2.Timer3.Enabled = True
  Case 5, 6, 7
    a$ = "deepr" + Trim(Str(k)) + ".bmp"
   Picture1.Picture=LoadPicture(a$)
    Picture2.Left = Form2.Width
  Case Else
  If MsgBox("Вы хотите закончить_
  обучение?", vbYesNo + vbQuestion,_
  "") = vbYes Then End
    k = 0
    Unload Form2
    Form1.Show
  End Select
End Sub
```

14. В процедуре обработки события Timer для Таймера 2 должны одновременно перемещаться Окна рисунков 2 и 3, содержащие изображение машины с прицепом. Движение осуществляется с шагом h справа налево до левой границы формы.

```
Private Sub Timer2_Timer()
  If Picture2.Left > 0 Then
    Picture3.Left = Picture3.Left - h
    Picture2.Left = Picture2.Left - h
  End If
End Sub
```

15. В процедуре обработки события Timer для Таймера 3 должно перемещаться справа налево Окно рисунка 2 с шагом h*5, содержащее изображение

Рисунок 12

Рисунок 13

мальчика с портфелем-«запятой», либо девочки с шариками в руках, в зависимости от текста-примера в Окне рисунка 1. Поддержка анимационного эффекта сменой рисунков-кадров производится с помощью переменной n, принимающей значение то равное нулю, то единице. Имена файлов с рисунками содержат, кроме порядкового номера (переменная k), также и номер кадра (переменная n), например, «Ris4-0.bmp» и «Ris4-1.bmp» (рисунки 12 и 13).

```
Private Sub Timer3_Timer()
   If Picture2.Left > 0 Then
      Picture2.Left = Picture2.Left - h * 5
      n = (n + 1) Mod 2
      a$ = "Ris" + Trim(Str(k)) + _
      "-"  + Trim(Str(n)) + ".bmp"
     Picture2.Picture = LoadPicture(a$)
   End If
End Sub
```

16. После написания кода программы требуется запустить ее на выполнение и окончательно протестировать.

Задание 5. Дальнейшая разработка обучающей программы по русскому языку (для самостоятельного выполнения).

Доработать приложение, представляющее собой обучающую программу по русскому языку по теме «Деепричастие и деепричастный оборот», рассмотренное в задании 4. Имеются дополнительные сведения для представления в качестве учебного материала:

1. Деепричастие имеет признаки глагола и наречия.

2. Деепричастия бывают совершенного и несовершенного вида.

3. Частица НЕ с деепричастиями пишется раздельно, существуют исключения.

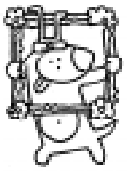
4. Имеются глаголы, от которых нельзя образовать деепричастий.

5. Существует замена деепричастному обороту – сложносочиненные и сложноподчиненные предложения.

Указания:

- создать в графическом редакторе рисунки, отражающие данную тему, разработать интерфейс пользователя с программой, используя одну или несколько форм, а также различные объекты управления и контроля;
- разработать сценарий, то есть последовательность действий пользователя и последовательность событий.

ОБЪЕКТ УПРАВЛЕНИЯ И КОНТРОЛЯ FRAME



Объект ***Frame (Рамка)*** предназначен для помещения в него других объектов, или, как говорят, служит *контейнером* для других элементов управления. Изменение значения свойств объекта-контейнера будет влиять на соответствующие свойства всех составляющих его объектов. Чтобы поместить объекты в рамку (группу), необходимо проследить, чтобы перед выбором других объектов рамка находилась в активном состоянии. Нельзя поместить в группу уже существующие на форме элементы управления.

Основные свойства:

Caption, BorderStyle, Enabled, FontBold, FontItalic, FontName, FontSize, Height, Left, Top, Width, Name – свойства, аналогичные свойствам других объектов.

СвойствоVisible, установленное в False, приведет к тому, что сама рамка и входящие в нее объекты исчезнут с экрана.

ОБЪЕКТ УПРАВЛЕНИЯ И КОНТРОЛЯ LISTBOX

Объект ListBox (Список) используется обычно в программах для наглядного представления информации, а также для предоставления пользователю возможности выбора одного (или нескольких) элемента из некоторого перечня.

Ниже перечислены основные элементы списка (рисунок 14).

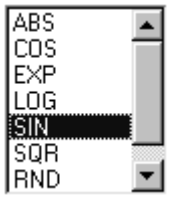


Рисунок 14

- Список элементов.
- Выбранный элемент (либо подсвечивается, либо помечается флажком в зависимости от типа списка).
- Полоса прокрутки.

Основные свойства:

Enabled, FontBold, FontItalic, FontName, FontSize, Height, Left, Top, Width, Name – свойства, аналогичные свойствам других объектов.

Columns (колонки) – количество колонок в списке, по умолчанию, равно нулю, что соответствует одной колонке.

List (список) – представляет собой массив, состоящий из элементов списка.

ListCount – количество элементов в списке.

ListIndex (индекс текущего элемента в списке) – соответствует номеру последнего подсвеченного элемента списка. Индекс первого элемента списка равен нулю.

Sorted (сортировка). Если это свойство приравнено True, то элементы в списке располагаются по алфавиту.

Style (вид списка). По умолчанию, равен нулю, при задании значения 1 список приобретает вид с элементами-флажками.

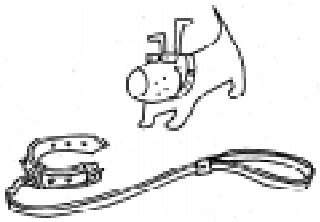
Основные методы:

AddItem – включает элемент в список (синтаксис: List1.AddItem текст[,индекс]).

RemoveItem – удаляет элемент из списка (синтаксис: List1.RemoveItem индекс).

Clear – удаляет из списка все элементы. (синтаксис: List1.Clear).

ДОПОЛНИТЕЛЬНЫЕ ОБЪЕКТЫ УПРАВЛЕНИЯ И КОНТРОЛЯ



При создании Visual Basic были предусмотрены специальные средства для расширения набора элементов управления и контроля, доступных программе. Для этого Вы можете выбрать пункт меню ***Components*** (контекстное меню Окна инструментария ***Toolbox***, либо в пункте ***Project*** главного меню). В появившемся списке вкладки Controls поставьте галочку против нужного элемента управления. Дополнительные стандартные элементы управления и контроля расположены в следующих группах:

- Microsoft Windows Common Controls 6.0 (содержит, например, такие объекты, как ImageList – Список изображений, StatusBar – Строка состояния, TreeView – Дерево просмотра, ProgressBar – Индикатор процесса и другие);
- Microsoft Windows Common Controls-2 6.0 (содержит такие объекты, как UpDown – Счетчик, MonthView – Календарь);
- Microsoft Windows Common Controls-3 6.0 (содержит объект CoolBar – Панель инструментов).

Кроме стандартных компонент, можно подключать к своим разработкам и другие интересные компоненты, с некоторыми из которых Вы познакомитесь далее.

ИСПОЛЬЗОВАНИЕ ТЕХНОЛОГИИ MICROSOFT AGENT

Microsoft Agent – средство, которое позволяет использовать анимированных персонажей (типа помощников в пакете

Microsoft Office) для создания оригинальных интерфейсных форм Ваших приложений. Microsoft Agent входит в состав операционной системы Windows-2000. Основной компонент ActiveX, персонажи (Characters) и синтезатор речи (text-to-speech, TTS) находятся в свободном распространении на http://www.microsoft.com/products/msagent.

Персонажи можно выбрать из галереи на http://allcharacters.chat.ru. С их помощью можно проводить мультимедийные презентации, озвучивать текст (в том числе и по-русски) сообщений в Ваших программах, забавной анимацией скрасить обучающую программу, протянуть руку помощи в тестирующей программе. При этом можно учесть психологические аспекты общения пользователя с Вашей программой, выбирая соответствующий тип персонажа: «Джина» (Genie) – покорного слугу Ваших действий или «Мага» (Merlin), чье могущество позволяет преодолеть преграды на пути познания. Установка дополнительных компонент: SAPI (Speech API) и BalloonDialog (http://www.sommytech.com.ar/balloondialog) обеспечит общение с агентом голосовыми командами и интерфейсом помощника MS Office. Дополнительная информация содержится в электронной документации VB по MS Agent.

АГЕНТ: – ДАВАЙТЕ ПОЗНАКОМИМСЯ

Создадим программу для первого знакомства с Агентом. Первая часть **–** *визуальное программирование*.

1. Поместим компонент Агента (после установки его основных модулей) в Toolbox – стандартный блок инструментов VB. Для этого, выберем опцию Components в меню Project. В появившемся списке вкладки Controls поставим галочку против элемента управления Microsoft Agent Control 2.0.

2. Добавим появившийся значок Агента на форму. Данный объект Agent1 содержит коллекцию объектов Characters для управления отдельным персонажем (может поддерживаться одновременно несколько персонажей с их взаимодействием). Каждый объект Character определяется строкой-именем и содержит объекты Commands (Команды) и Balloon (специальное окно для вывода текстовой информации). Объект Commands служит для распознавания отдельным персонажем как голосовых команд, так и выбираемых из контекстного меню, которое вызывается по нажатию правой кнопки мыши на персонаже. Каждый персонаж задается отдельным файлом (с расширением *.acs) и имеет своеобразный набор анимации. На рисунке 15 в качестве Агента выбран Бобик (Rocky.acs) – помощник MS Office. Для синхронизации действий агента и других операторов программы служит объект Request, позволяющий получать информацию о состоянии выполнения Агентом различных методов.

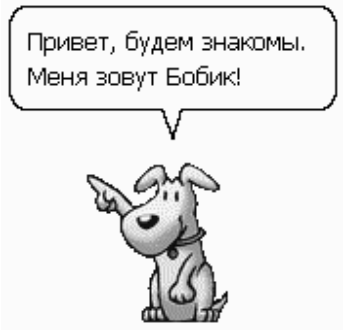


Рисунок 15

3. Поместим на форму Командную кнопку 1 с надписью «Вызвать агента» и Командную кнопку 2 с надписью «Список команд». Для содержимого списка используем элемент управления ListBox.

Вторая часть **–** *написание кода программы***.**

4. Используем глобальную переменную MyAgent для обозначения персонажа.

```
Dim MyAgent As IAgentCtlCharacterEx
```

5. В процедуре обработки события Щелчок на кнопке 1 выполним следующие действия:

- Загрузим (Load) файл персонажа в коллекцию Characters под именем «merlin» (файл «merlin.acs» или «genie.acs» или «rocky.acs» или какой-либо другой, присутствующий в системе).
- Добавим с помощью метода Add в раздел «Мои команды» пользовательские команды (в том числе и голосо-

вые при установленном SAPI).

· Переместим без анимации персонаж в заданную точку экрана (Move).

· Покажем персонаж (Show), текст (Speak) и анимацию его приветствия (Play).

```
Private Sub Command1_Click()
Agent1.Characters.Load "merlin",_
"merlin.acs"
Set MyAgent = Agent1.Characters("merlin")

MyAgent.Commands.Caption = "Мои_
команды"
MyAgent.Commands.Voice = "Voice_
commands"
MyAgent.Commands.Add "Cmd1",_
"Запустить Explorer", "Explorer"
MyAgent.Commands.Add "Cmd2",_
"Спрятаться в углу", "Move"
MyAgent.Commands.Add "Cmd3",_
"Давайте знакомиться!"
MyAgent.Commands.Add "Options",_
"Параметры агента", "Options"
MyAgent.Commands.Add "Sing", "Спой",
"Spoi", True, True

MyAgent.MoveTo 450, 120, 0
MyAgent.Show
'Для выбора приветствия в случайном_
порядке используется знак "|"
MyAgent.Speak("Hello,_
friend!|Greetings!|Here I am!")
MyAgent.Play "Wave"
MyAgent.Play "RestPose"
End Sub
```

6. В процедуре обработки события Щелчок на кнопке 2 делаем видимым список List1 и добавляем в него всевозможные для данного персонажа анимации.

```
Private Sub Command2_Click()
List1.Visible = True
For Each Animation In_
MyAgent.AnimationNames
  List1.AddItem Animation
Next
End Sub
```

7. В процедуре обработки команд агента определяем его действия по пользовательским командам с именами:

· Cmd1 – запустить программу Explorer.

· Cmd2 – переместить программно агента в верхний левый угол (голосовая команда «move»).

· Cmd3 – показать в Balloon-окне стандартное имя и описание агента.

· Options – показать окно дополнительных параметров персонажа, в котором можно выбрать клавишу (по умолчанию – CapsLock), по нажатию которой персонаж переходит в режим прослушивания микрофона в течение задаваемого времени (в настройке – 2 секунды), когда команда может быть озвучена голосом (третий параметр метода Commands.Add).

· Sing – персонаж проигрывает заданный wav файл (голосовой аналог команды «spoi»).

```
Private Sub Agent1_Command(ByVal_
UserInput As Object)
  Select Case UserInput.Name
    Case "Cmd1"
      MyAgent.Think ("Выполняю_
      команду")
      Shell "explorer", vbNormalFocus
    Case "Cmd2"
      MyAgent.MoveTo 1, 1
    Case "Cmd3"
      MyAgent.Speak MyAgent.Name +_
      ". " + MyAgent.Description
    Case "Options"
     Agent1.PropertySheet.Visible_
     = True
    Case "Sing"  'merlin "говорит"_
    звуковой файл
      Path$ = "c:\windows\media_
      \crialarm.wav" 'укажите_
      полный путь к файлу
      MyAgent.Speak "", Path$
  End Select
End Sub
```

8. В процедуре обработки события Щелчок на элементе списка List1 персонаж останавливает свои предыдущие действия и выполняет заданную анимацию (аналог команды «Мотор!» из локального меню помощника MS Office).

```
Private Sub List1_Click()
  MyAgent.Stop
  MyAgent.Play List1.Text
End Sub
```

9. Запустите проект, вызовите соответствующей кнопкой агента и список анимации. Пронаблюдайте действия агента для каждого элемента списка (они мо-

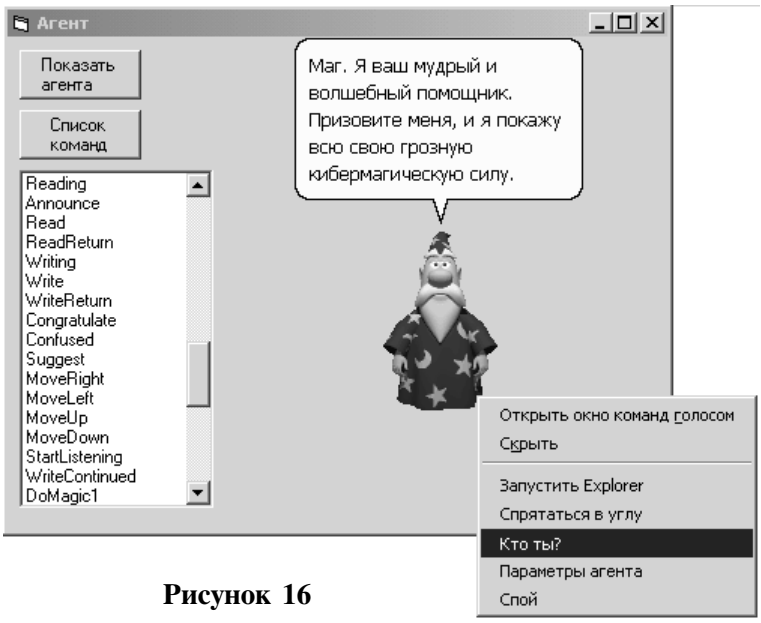
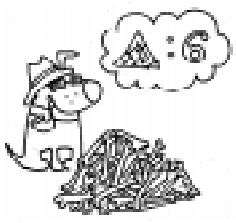


Рисунок 16

гут быть разными даже для одного элемента списка). Вызовите контекстное меню агента для тестирования собственных и стандартных команд. В режиме прослушивания агентом опробуйте голосовые команды. В случае возникновения ошибок, когда выбранным агентом не поддерживаются определенные методы, закомментируйте соответствующие операторы. На рисунке 16 представлен фрагмент работы программы.

Задание 6. Обучающая программа по математике.



Разработайте обучающую программу, которая демонстрировала бы метод решения линейных уравнений вида $\mathbf{a}\,x + \mathbf{b} = \mathbf{c}\,x + \mathbf{d}$. Используйте технологию Microsoft Agent, чтобы придать приложению динамику и колорит. Пусть агент сделает Вашу обучающую программу наглядной и увлекательной.

Выполнение задания 6.

Метод решения можно продемонстрировать на конкретном примере:

5x – 20 = 10 + 2x.

Первая часть – *визуальное программирование*.

1. Создайте форму, как на рисунках 17 и 18, придайте ей нужные размеры и заголовок.

2. Разместите на форме рамку Frame1 без границы (BorderStyle = 0) и сделайте ее невидимой (Visible = False).

3. Внутри этой рамки расположите по порядку метки e1, e2, e3, e4, e5 с заголовками «5x», «– 20», «=», «10» и «+ 2x», соответственно.

4. Ниже добавьте еще одну невидимую рамку Frame2, куда поместите 4 отрезка (см. рисунок 18) и метки «3x = 30», «:3». Надпись «:3» выделите красным цветом, а соответствующую ей метку назовите d3 и сделайте невидимой – в дальнейшем она нам понадобится.

5. Еще ниже расположите метку Final, где буден высвечиваться ответ «x = 10». Спрячьте ее, изменив свойство Visible. Спрятанные объекты будут по очереди показываться во время ведения урока.

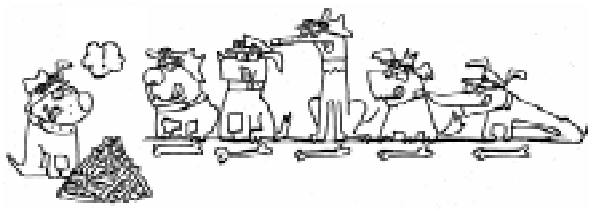
6. Наконец, добавьте на форму объект Agent. Если такого компонента нет на панели инструментов, отметьте галочкой Microsoft Agent Control в меню Project и Components.

Вторая часть – *написание кода программы*.

7. Для начала определим переменную Wizard, которую в программе будем использовать для обращения к агенту – персонажу, ведущему урок.

```
Dim Wizard As IAgentCtlCharacterEx
```

8. Поскольку персонажи MS Agent говорят и действуют независимо (параллельно исполнению программы), то пона-



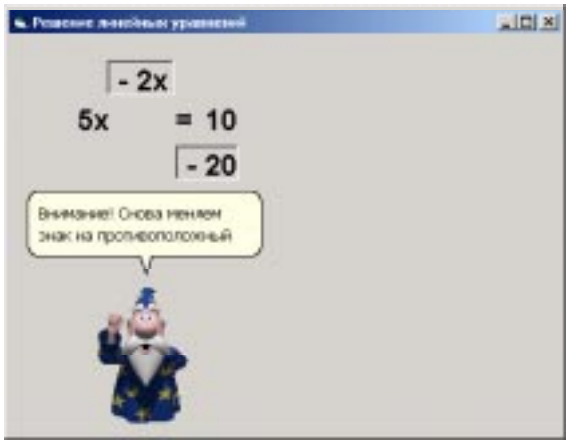


Рисунок 17

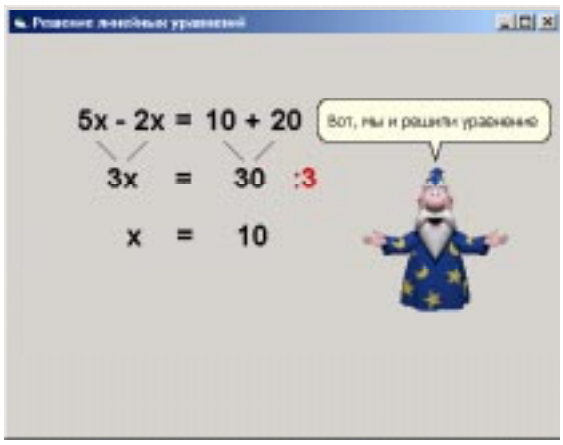


Рисунок 18

добится синхронизировать поведение агента, то есть приостанавливать выполнение программы до тех пор, пока агент не закончит свою речь или анимацию. Для этой цели будет служить процедура Wait, параметром которой является объект IAgentCtlRequest, порождаемый любым действием агента (Speak, Play, Move и т.п.). У этого объекта есть свойство Status, которое не равно нулю, пока агент исполняет команду, и обращается в нуль при завершении действия. Во время ожидания персонажа целесообразно передавать управление операционной системе Windows для обработки событий, чтобы программа не «зависала», а продолжала реагировать на действия пользователя. Для этого предназначена стандартная функция DoEvents.

```
Sub Wait(Action As IAgentCtlRequest)
   Do
      DoEvents
   Loop Until Action.Status = 0
End Sub
```

9. Кроме того, для анимации (мигания или перемещения объектов) весьма полезной окажется процедура Sleep, приостанавливающая исполнение программы на некоторый промежуток времени, продолжительность которого будет передаваться в качестве параметра Interval. Процедуру легко реализовать при помощи функции Timer, возвращающей количество секунд, прошедших с начала суток. Достаточно организовать цикл, работающий до тех пор, пока значение Timer не увеличится на заданный интервал. Во избежание «зависания» программы, опять же поместим внутрь цикла вызов DoEvents.

```
Sub Sleep(Interval As Single)
   StartTimer = Timer
   Do
      DoEvents
   Loop Until Timer - StartTimer >=_
Interval
End Sub
```

10. Во время урока будет наглядно продемонстрирована процедура перемещения неизвестных членов в левую часть уравнения, а известных – в правую. Для этого потребуется передвигать по форме соответствующие им метки. Удобно создать две процедуры, реализующие перемещение некоторого объекта как по горизонтали, так и по вертикали. Параметрами этих процедур будут являться перемещаемый объект (в нашем случае метка, поэтому назовем этот параметр Label) и новая координата объекта. Процедура MoveX, передвигающая метку по горизонтали, будет изменять ее свойство Left, а MoveY, перемещающая объект по вертикали, – свойство Top.

```
Sub MoveX(Label, NewX As Integer)
   'при движении вправо шаг_
   положительный, влево – отрицательный
   If Label.Left < NewX Then s = 20_
   Else s = -20
   For i = Label.Left To NewX Step s
      Label.Left = i
```

```
    Sleep 0.05
  Next
End Sub

Sub MoveY(Label, NewY As Integer)
  'при движении вниз шаг_
  положительный, вверх – отрицательный
  If Label.Top < NewY Then s = 20_
  Else s = -20
  For i = Label.Top To NewY Step s
    Label.Top = i
    Sleep 0.05
  Next
End Sub
```

11. Дважды в программе будет продемонстрирована сцена с миганием знака «=», поэтому оформим этот эпизод в виде отдельной процедуры.

```
Sub BlinkSign()
  'несколько раз прячем и показываем_
  знак "=" с интервалом в полсекунды
  For i = 1 To 4
    e3.Visible = Not e3.Visible
    Sleep 0.5
  Next
End Sub
```

12. Поскольку ответные действия пользователя в программе не предусмотрены, не потребуется реализовывать событийные процедуры, кроме одной, Form_Activate, вызываемой в начале исполнения программы. В отличие от Load, событие Activate возникает уже после того, как форма будет показана на экране. Именно внутри этой процедуры запрограммируем по порядку весь ход урока от начала и до конца. Программа представлена в приложении 1.

Задание 7.
(для самостоятельного выполнения).

Разработайте обучающую программу с использованием Агента для объяснения произвольной темы из любой предметной области.

Дополнительный материал и ответы на вопросы по технологии MS Agent Вы можете получить через Интернет из конференции Microsoft.public.msagent.

Приложение 1.

```
Private Sub Form_Activate()
  'загружаем персонаж Merlin в коллекцию Agent.Characters
  'это будет волшебник, поэтому назовем его Wizard
  Agent.Characters.Load "Wizard", "Merlin.acs"
 'для удобства обращения присваиваем переменной Wizard ссылку на персонажа
  Set Wizard = Agent.Characters("Wizard")
  'помещаем волшебника в нужное место на экране и показываем его
  Wizard.MoveTo 440, 150
  Wizard.Show
  'далее агент приветствует пользователя и кланяется
  Wizard.Speak "Привет, друг!"
  Wizard.Play "Greet"
  Wizard.Play "RestPose"
  Wizard.Speak "Я расскажу тебе, как решаются линейные уравнения"

  'агент взмахивает волшебной палочкой…
  Wizard.Play "DoMagic1"
  Wait Wizard.Play("DoMagic2")
  '…после чего на экране появляется исходное уравнение
  Frame.Visible = True
  'агент представляет уравнение, показывая на него рукой
  Wizard.Play "GestureRight"
  Wizard.Speak "Вот, типичный пример линейного уравнения"
```

```
Wizard.Speak "Перенесем неизвестные члены в левую часть..."
'волшебник переносится под метку "+2x" и возносит руки вверх
Wizard.MoveTo 285, 225
Wait Wizard.Play("GestureUp")
'метка "+2x" обводится в рамку
e5.BorderStyle = 1
'затем перемещается сначала вверх на 520, затем влево до знака "="
MoveY e5, e5.Top - 520
MoveX e5, e3.Left

'перемещаем волшебника чуть левее, чтобы он не загораживал уравнение
Wizard.Left = 360
'заставляем его показать рукой на знак "=" и произнести фразу
Wizard.GestureAt 100, 50
Wait Wizard.Speak("При переносе членов через знак =_
 меняем знак на противоположный")
'после чего "=" мигает, а заголовок метки "+2x" меняется на "-2x"
BlinkSign
e5.Caption = " - 2x"
'продолжаем двигать метку влево до надписи "-20"
MoveX e5, e2.Left

Wizard.Speak "Теперь перенесем известные члены вправо..."
'заставляем агента "телепортироваться" под надпись "-20"
Wizard.Hide
Wizard.MoveTo 175, 260
Wizard.Show
'он снова поднимает руки
Wait Wizard.Play("GestureUp")
'а метка "-20" обводится в рамку и перемещается вниз и вправо до "="
e2.BorderStyle = 1
MoveY e2, e2.Top + 520
MoveX e2, e3.Left

'после того, как метка "доехала" до знака "=", волшебник сдвигается
'чуть вниз и, дважды постучавшись, привлекает внимание пользователя
Wizard.Top = 330
Wizard.Play "GetAttention"
Wizard.Play "GetAttentionContinued"
Wait Wizard.Speak("Внимание! Снова меняем знак на противоположный")
'снова мигает знак "=", и надпись "-20" заменяется на "+20"
BlinkSign
e2.Caption = "+20"

'известные, затем неизвестные члены расставляются по своим новым местам
MoveX e2, e4.Left + 500
MoveY e2, e3.Top
MoveY e5, e3.Top
'и рамка, обрамляющая их, исчезает
e2.BorderStyle = 0
e5.BorderStyle = 0

'волшебник перелетает вправо на свое прежнее место
Wizard.MoveTo 440, 190
Wizard.Speak "Следующим шагом приведем подобные члены"
```

```
  'и взмахивает волшебной палочкой
  Wizard.Play "DoMagic1"
  Wait Wizard.Play("DoMagic2")
  'показывается панель, на которой видно уравнение с приведенными членами
  Frame2.Visible = True
  Wait Wizard.Speak("Поделим обе части уравнения на коэффициент при x")
  'появляется метка с надписью ":3"
  d3.Visible = True
  'волшебник "думает" над ответом
  Wait Wizard.Play("Confused")
  'наконец, появляется и сам ответ
  Final.Visible = True

  'Финал: агент произносит заключительную фразу и трубит в горн
  Wizard.Play "Explain"
  Wizard.Speak "Вот, мы и решили уравнение"
  Wizard.Play "Announce"
End Sub
```

Паньгина Нина Николаевна,
преподаватель ОИиВТ
школы-лицея № 8, г. Сосновый Бор.

Паньгин Андрей Александрович,
студент 3 курса математико-
механического факультета СПбГУ.