Практическая работа №14, Угадай слово

Постановка задачи

Разработать программу, которая реализует логическую игру «Угадай слово», известную каждому с детства.

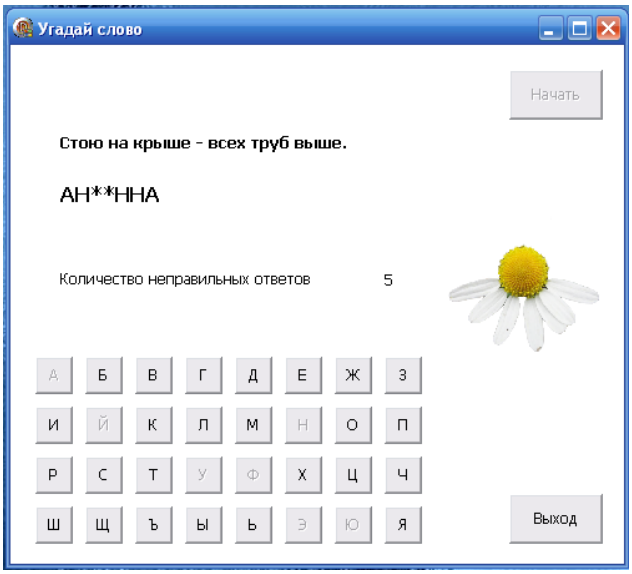


Рис.1

Правила игры

Ведущий загадывает слово, которое надо угадать, и записывает черточки, количество которых равно количеству букв в загаданном слове.
(Для подсказки можно использовать вопрос, ответом на который является загаданное слово.) Игроку предоставляется право выбора букв по одной, при этом он может допустить только десять ошибок (попыток).
Игрок называет букву, которая на его взгляд может присутствовать в слове. Если выбранная игроком буква есть в загаданном слове, то ведущий записывает ее вместо черточки на своем месте. Если буква встречается в слове несколько раз, то ведущий «открывает» все эти буквы. Если буква названная игреком отсутствует в слове, то уменьшается количество предоставленных попыток.
Игра заканчивается, если слово угадано, или если исчерпаны все предоставленные попытки.

Новым в этой работе являются:

- создание компонент во время выполнения программы и обработка их событий,
- вывод иллюстраций.

Информационная постановка задачи

1. Информация о вопросах и ответах хранится в текстовом файле (Questions.txt). На каждый вопрос-ответ отводится две строки: в первой строке - вопрос, а во второй - ответ.
2. Информация из текстовых файлов в начале программы считывается в массив (AQ), из которого затем случайным образом выбираются тексты вопросов.
3. Выбранный вопрос выводится на экран, а ответ записывается в переменную STR_R.
4. Формируется угадываемое слово в виде черточек (переменная STR_N). Количество черточек соответствует количеству букв (например, STR_N='------').

5. Для выбора отгадываемых букв слова используются кнопки, которые создаются в ходе программы.
6. При нажатии на кнопку с буквой, которая есть в слове, эта буква появляется на нужном месте, т.е. меняется переменная STR_N (например, STR_N='-A-A--'). Если такой буквы нет, то уменьшается количество попыток (переменная ATTEMPT). Кнопка, содержащая нажатую букву, в дальнейшем она становится недоступной.
7. При угадывании слова (STR_R = STR_N) игроку предоставляется возможность начать новую игру или выйти из игры.
8. При использовании всех попыток (ATTEMPT=0) на экране появляется неугаданное слово и предоставляется возможность начать новую игру или выйти из игры.

План разработки программы

Построение программы будем поэтапно.

Первый этап. Формирование формы экрана

1. Откройте новый проект.

2. Разместите в форме экземпляры компонентов в соответствии с рис.2.

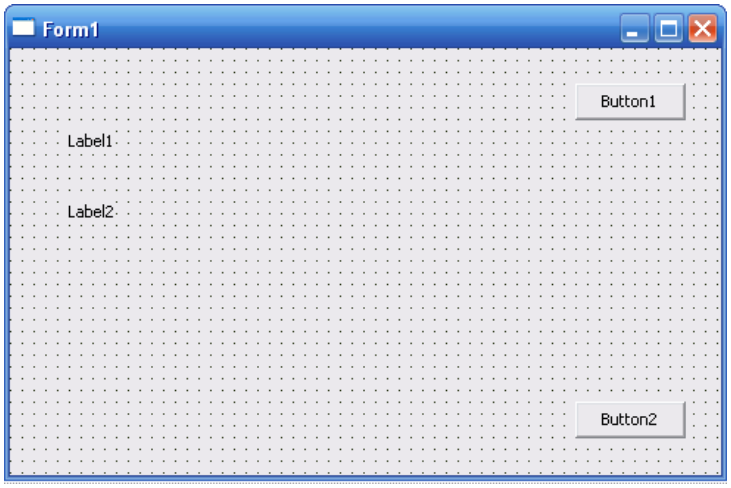


Рис.2

3. Выделите объект **Form1**, перейдите на вкладку **Events Инспектора объектов** (**Object Inspector**), найдите событие **OnCreat**, справа от него дважды щелкнуть левой кнопкой мыши. Попав в код программы, надо написать следующий текст:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
// Формирование элементов формы
  With Form1 do
  begin
   Caption:='Угадай слово';
   Height:=450;
   Width:=500;
   Color:=clWhite;
  end;
  With Button1 do
  begin
    Caption:='Начать';
    Height:=40;
    Width:=75;
    Top:=20;
    Left:=400;
    Font.Size := 9;
```

```
  end;
  With Button2 do
  begin
    Caption:='Выход';
    Height:=40;
    Width:=75;
    Top:=360;
    Left:=400;
    Font.Size := 9;
  end;
  With Label1 do
  begin
    Caption:='';
    Height:=40;
    Width:=250;
    Top:=70;
    Left:=40;
    Font.Style:=[fsBold];
    Font.Size := 10;
  end;
  With Label2 do
  begin
    Caption:='';
    Height:=40;
    Width:=75;
    Top:=110;
    Left:=40;
    Font.Style:=[fsBold];
    Font.Size := 12;
  end;
 Read_File; //Процедура чтения информации из файла
end;
```

Пояснение

Для компонент **Form1**, **Button1**, **Button2**, **Label1**, **Label2** в тексте программы определены все необходимые свойства (размеры, местоположение и т.п.), поэтому при размещении компонент нет необходимости устанавливать свойства компонент с помощью вкладки **Properties** (Свойства) инспектора объектов.

4. Разместите в блоке реализации после слова **implementation** описание констант и переменных:

```
Const
 N=100;   //Максимальное количество записей в массиве
TYPE
 T_R = record     //Структура записи массива
        Que:string[250];
        Ans:string[30];
       end;
 R=array[1..N]of T_R;

Var AQ:R;                    //Массив Вопросов и Ответов
    Quetions_F:TextFile;     //Файловая переменная
    STR_R,STR_N:String[30]; //Отгадываемое слово
    KOL_QUE:Integer;         //Количество вопросов в файле
```

5. Разместите после блока описания переменных процедуру чтения информации из файла и формирование массива вопросов **AQ**. К этой процедуре происходит обращение в конце процедуры **TForm1.FormCreate** (см.п.3).

```
procedure Read_File;
//Чтение информации из файла и запись в массив
Var KOL, I:Integer;
begin
 Assignfile(Quetions_F,'Quetions.txt');
 Reset(Quetions_F);
 KOL:=1; I:=1;
 While not Eof(Quetions_F) do
  begin
   if (KOL mod 2)=1 then Readln (Quetions_F, AQ[I].QUE)
                    else
                    begin
                     Readln(Quetions_F, AQ[I].ANS);
                     Inc(I);
                    end;
   Inc(KOL);
  end;
 KOL_QUE:=I;
 closefile(Quetions_F);
end;
```

6. Создадим процедуру, которая обрабатывает ситуацию нажатия кнопки **Button1** (Начать игру). Для этого выделите объект **Button1**, перейдите на **вкладку Events Инспектора объектов** (**Object Inspector**), найдите событие **On Click**, справа от него дважды щелкнуть левой кнопкой мыши. Попав в код программы, надо написать следующий текст:

```
procedure TForm1.Button1Click(Sender: TObject);
Var I,NUMBER:integer;
begin
 Randomize;
 NUMBER:=Random(KOL_QUE-1)+1;
 Label1.Caption:=AQ[NUMBER].QUE;
 STR_R:=AQ[NUMBER].ANS;
 STR_N:='';
 for I := 1 to Length(STR_R) do
 STR_N:=STR_N+'*';
 Label2.Caption:=STR_N;
 Form1.Button1.Enabled:=False;
end;
```

Пояснение

Начало игры означает, что нужно выбрать вопрос, который затем выводиться на экран (**Label1**), и сформировать зашифрованное слово (**Label2**, **STR_N**), которое будет отгадывать игрок. Выбор вопроса из массива производится с помощью функции Random.

7. Самостоятельно добавьте событие обработки кнопки **Button2** (Выход).

8. Создайте текстовый файл **Quetions.txt**, записав его в ту же папку, где находится и программа. Текст файла может выглядеть так:

Стою на крыше - всех труб выше.

АНТЕННА

Всех на свете обшивает, а сама не надевает.

ИГОЛКА

Твой хвостик я в руке держал, ты полетел, я побежал.

ШАРИК

Золотое решето, черных домиков полно.

ПОДСОЛНУХ

Ах, не трогайте меня. Обожгу и без огня!

КРАПИВА

9. Сохраните проект, запустите и протестируйте его. После запуска программы на экране видны две кнопки - **Начать** и **Выход**. Если нажать кнопку **Начать**, то появляется текст вопроса и зашифрованное слово ответа, при этом кнопка **Начать** становится недоступной. Проверьте, чтобы при каждом новом запуске программы у вас всегда выбирался новый текст ответа.

Второй этап. Создание компонент во время выполнения программы и обработка их событий

Для дальнейшей работой над проектом нам необходимо создать 33 кнопки, которые будут принимать значения букв русского алфавита, и обрабатывать ситуацию поиска выбранной буквы (нажатой кнопки). Эта задача не сложная, но очень утомительная – 33 раза повторить одни и те же операции. Как упростить этот процесс покажем на отдельном примере.

В данном примере по нажатию кнопки **Button1** создаются 32 кнопки, которые располагаются в три ряда. Свойства **Caption** этих кнопок принимают значения букв русского алфавита от «А» до «Я» (кроме буквы «Ё»). По нажатию каждой кнопки идет обращение к процедуре **TForm1.BtnClick**, которая определяет какая буква русского алфавита соответствует нажатой кнопке. Дальше идет обращение к процедуре **Poisk**, и нажатая кнопка становится недоступной.

Процедура **Poisk** ищет встречается ли выбранная буква (нажатая кнопка) в заданном слове **STR_N**, и если да, то заменяется в **STR_R** соответствующий символ на заданную букву.

1. Откройте новый проект.

2. Разместите в форме экземпляры компонентов **Label1** и **Button1**.

3. Для события **OnClick** компоненты **Button1** напишите такой текст:

```
procedure TForm1.Button1Click(Sender: TObject);
var
 i: integer;
begin
for i:=1 to N do           // Цикл по созданию кнопок на форме
begin
  Btn:=TButton.Create(Form1);
  with Btn do
   begin             // Определение свойств создаваемых кнопок
    Parent := Form1; // Определение родителя,
                     //т.е. где создаются кнопки
    Caption := Chr(191+I);//Определение буквы по ее коду
    Height := 30;
    Width := 30;
    Top := 200+((I-1) div 11)*40;
    Left := ((I-1) mod 11 ) * 40 + 30;
    Font.Style:=[fsBold];
    OnClick := FORM1.BtnClick;  // Имя процедуры,
                       //обрабатывающей событие нажатия кнопки
  end;
```

```
 end;
  STR_R:='РОССИЯ'; //Для примера наше отгадываемое слово
  STR_N:='';
  For I:=1 to Length(STR_R) do
  STR_N:=STR_N+'*';
  Label1.Caption:=STR_N;
  Form1.Button1.Enabled:=False;
 end;
```

4. Процедуру **`TForm1.BtnClick`** (обработка события нажатия созданных кнопок) мы создаем сами. Для этого вставьте текст этой процедуры в общий текст программы перед тем, как осуществляется обращение к процедуре **`TForm1.BtnClick`**.

```
procedure TForm1.BtnClick(Sender: TObject);
// Процедура обработки события нажатия созданных кнопок
Var STR_:String;
begin
STR_:= (Sender as TButton).Caption;
// Переменная Sender содержит имя
// объекта, которому соответствует данное событие
CHAR_:=STR_[1];
POISK(STR_R,STR_N,CHAR_); //Обращение процедуре поиска буквы
(Sender as TButton).Enabled:=False; //Кнопка буквы недоступна
end;
```

5. В раздел описания процедур (перед **`private`**) добавьте строку описания заголовка процедуры:

```
procedure BtnClick(Sender: TObject);
```

6. Для того, чтобы наша программа заработала необходимо внести объявление переменных и описание процедуры **Poisk**.

```
 Const N=32;              //Количество букв-кнопок
 Type STR_30=String[30];
 Var Btn: TButton;        // Переменной для создания кнопок
     STR_N,               // Зашифрованное слово
     STR_R:STR_30;,       // Отгадываемое слово
     CHAR_:Char;          // Нажатая буква
   {$R *.dfm}

Procedure POISK(STR_R:STR_30;Var STR_N:STR_30;CHAR_:Char);
//Поиск буквы в слове и «открытие» ее
Var I:Integer;
      Flag:Boolean;     // Флаг найдена ли в слове нажатая буква
  begin
  Flag:=False;
  For I:=1 to Length(STR_R) do  // Поиск буквы в слове
   If STR_R[I]=CHAR_ then
     begin
      STR_N[I]:=CHAR_;
      Flag:=True;
     end;
  If Flag=True Form1.Label1.Caption:= STR_N;

  end;
```

7. Сохраните проект, запустите и протестируйте его. После запуска программы на экране видна только одна кнопка **Button1** и закодированное слово в виде звездочек. После нажатия на эту кнопку на экране появляется три ряда кнопок с буквами (см.рис.3) и кнопка **Button1** становится недоступной.

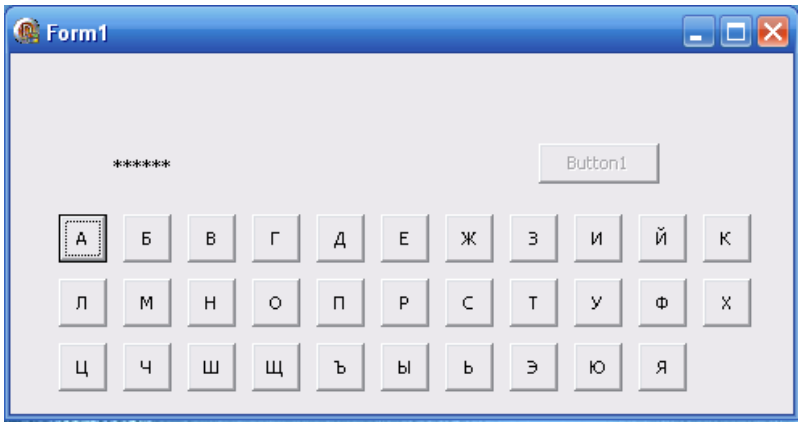


Рис.3

Нам известно слово, которое зашифровано – это слово «РОССИЯ». Протестируйте программу, введя буквы этого слова, и проверьте, чтобы они у вас «открылись» в зашифрованном слове.

Третий этап. Создание кнопок перед началом игры

Выполните этот этап самостоятельно, внеся изменения в основной текст программы, созданный на втором этапе. При создании кнопок расположите их не в три ряда, а в четыре (см.рис.4). Подумайте, как это сделать.

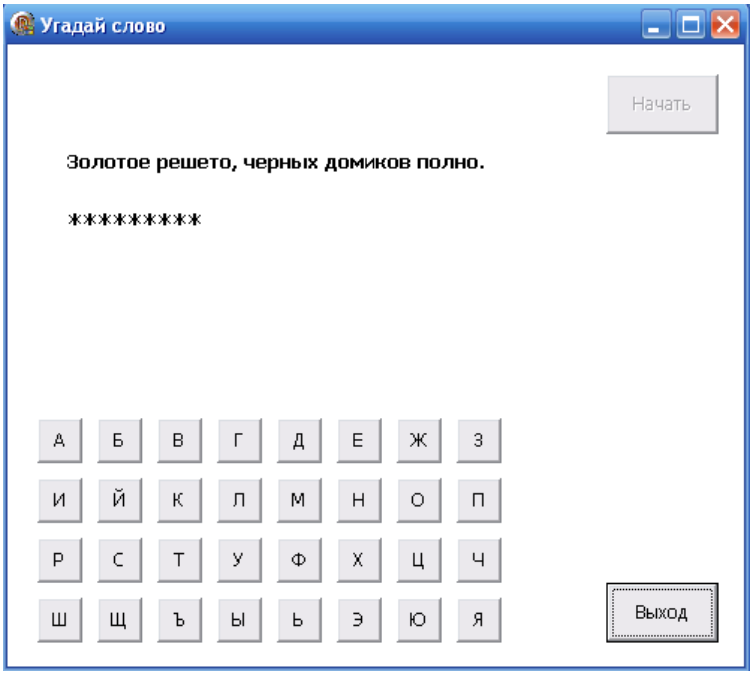


Рис.4

Четвертый этап. Анализ состояния игры

Необходимо внести изменения в программу, для определения закончена ли игра, т.е. угаданы все буквы в зашифрованном слове. Вместе с тем установим количество допустимых ошибок игрока.

1. Разместите в блоке реализации описание дополнительных констант и переменных:

```
Const
  Attempt=10; //Максимальное количество попыток
```

```
Var N_Attempt:Integer;  //Количество ошибок
```

2. Разместите в форме экземпляры компонентов **Label3, Label4** (сообщение о количестве допущенных ошибок) и **Panel1** (сообщение о завершении игры).

3. Дополните текст процедуры **TForm1.FormCreate** описанием новых компонент:

```
With Label3 do
  begin
    Caption:='';
    Height:=40;
    Width:=75;
    Top:=180;
    Left:=40;
    Font.Size := 9;
  end;
  With Label4 do
  begin
    Caption:='';
    Height:=40;
    Width:=75;
    Top:=180;
    Left:=300;
    Font.Style:=[fsBold];
    Font.Size := 10;
  end;
  With Panel1 do
  begin
    Caption:='';
    Height:=45;
    Width:=185;
    Top:=15;
    Left:=110;
    Font.Style:=[fsBold];
    Font.Size:= 10;
    Visible:=False;
  end;
```

4. Вначале процедуры **TForm1.Button1Click** добавьте следующий текст:

```
 Form1.Panel1.Visible:=False;
 Label3.Caption:='Количество неправильных ответов';
 N_Attempt:=0;
 Label4.Caption:=IntToStr(N_Attempt);
```

5. Внесите изменения в текст процедуры поиска POISK для анализа результата. Процедура может выглядеть следующим образом:

```
Procedure POISK(STR_R:STR_30;Var STR_N:STR_30;CHAR_:Char);
//Поиск буквы в слове и «открытие» ее
Var I:Integer;
Flag:Boolean; // Флаг найдена ли в слове нажатая буква
begin
Flag:=False;
For I:=1 to Length(STR_R) do begin// Поиск буквы в слове
  If STR_R[I]=CHAR_ then
   begin
    STR_N[I]:=CHAR_;
    Flag:=True;
   end;
```

```
 end;
If Flag=False Then
 begin N_Attempt:=N_Attempt+1;
  Form1.Label4.Caption:=IntToStr(N_Attempt);
 end
 Else Form1.Label2.Caption:= STR_N;
If STR_N=STR_R then   //Игра окончена - отгадано все слово
 begin
 With Form1.Panel1 do
  begin
   Visible:=True;
   Font.Color:=clBlack;
   Caption:='Вы выиграли!';
  end;
   Form1.Button1.Enabled:=True;
  end;
 If N_Attempt=Attempt then
//Игра окончена - количество ошибок равно количеству попыток
 begin
  With Form1.Panel1 do
   begin
    Visible:=True;
    Font.Color:=clRed;
    Caption:='Вы проиграли...';
   end;
  Form1.Button1.Enabled:=True;
  end;
end;
```

6. Сохраните проект, запустите и протестируйте его.

Пятый этап. Вывод иллюстраций

Для того, чтобы программа была более привлекательной, добавим вывод измененных изображений в зависимости от количества допущенных ошибок. Ряд изменения изображений может выглядеть так:




0 ошибок 1 ошибка 2 ошибки 3 ошибки и т.д.

Осталось только добавить эти изображения в программу, но сначала создадим самостоятельную программу, которая будет выводить изображения в зависимости от нажатия кнопки.

Немного теории

Для вывода иллюстрации используется компонент **Image**, который находится на вкладке **Additional** палитры компонентов.

В таблице приведены основные свойства компонента **Image**.

Имя свойства	Значение
Name	Имя компонента
Picture	Свойство, являющееся объектом типа **Tbitmap**. Определяет выводимую картинку
Left	Расстояние от левого края формы до левой границы области картинки
Top	Расстояние от верхней границы формы до верхней границы области картинки
Height	Высота картинки
Width	Ширина картинки
Stretch	Признак автоматического сжатия или растяжения картинки таким образом, чтобы она была видна полностью в области, размер которой задан свойствами **Width** и **Height**
AutoSize	Признак автоматического изменения размера компонента в соответствии с реальным размером картинки. Если значение свойства **AutoSize** равно **True**, то при изменении значения свойства Picture автоматически меняется размер области вывода иллюстрации так, чтобы была видна вся картинка. Если значение свойства **AutoSize** равно **False**, а размер картинки превышает размер области, то отображается только часть картинки

Картинку, отображаемую в области **Image**, можно задать во время создания формы или во время работы программы:

- Во время создания формы картинка задается установкой значения свойства **Picture**.
- Во время работы программы – применением метода **LoadFromFile**.

Например, для вывода иллюстрации, находящейся в файле **dog.bmp**:

```
Image1.Picture.LoadFromFile('dog.bmp');
```

При проектировании формы можно жестко задать предельный размер иллюстрации. И если реальный размер иллюстрации превышает размер области, выделенной для ее вывода, то необходимо вычислить коэффициент масштабирования и установить максимально возможные, пропорциональные ширине и высоте иллюстрации, значения свойств **Width** и **Height** области вывода иллюстрации. А если размер иллюстрации меньше области вывода, то можно пропорционально увеличить картинку.

Реальные размеры иллюстрации, загруженной в область **Image**, можно получить из свойств:

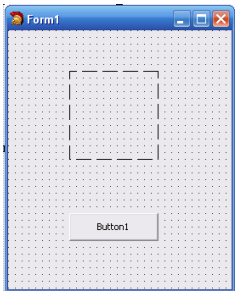
```
Image1.Picture.Bitmap.Width
Image1.Picture.Bitmap.Height.
```

Пример программы вывода картинок

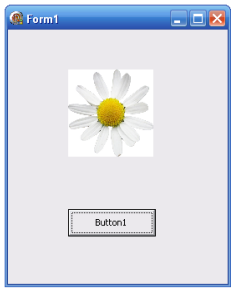
Программа позволяет последовательно выводить на экран при нажатии на кнопку **Button1** три картинки, находящиеся в файлах **1.bmp**, **2.bmp**, **3.bmp**. После последней картинки

будет выведена опять первая - **1.bmp**. При выводе на экран размер картинки масштабируется в зависимости от заданного размера компонента **Image**.

Откройте новый проект. Разместите на форме экземпляры компонентов **Image** и **Button**.



Исходная форма



Выполнение программы

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    Image1: TImage;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
Var
   iw,ih: integer;   // Первоначальный размер компонента Image
   N_Image: integer; // Номер выводимой картинки
   AFile: String;    // Имя картинки

{$R *.dfm}
// изменение размера области вывода иллюстрации пропорционально
// размеру иллюстрации
Procedure ScaleImage;
// Масштабирование изображения
var
    pw, ph : integer;       // Размер иллюстрации
    scaleX, scaleY : real; // Масштаб по X и Y
    scale : real;          // Масштаб
begin
      // Иллюстрация уже загружена, получаем ее размеры
```

```
      pw := Form1.Image1.Picture.Width;
      ph := Form1.Image1.Picture.Height;
      scaleX := iw/pw;
      scaleY := ih/ph;

      // Выбераем наименьший коэффициент
      if  scaleX < scaleY
          then scale := scaleX
          else scale := scaleY;

      // Изменяем размер области вывода иллюстрации
Form1.Image1.Height := Round(Form1.Image1.Picture.Height*scale);
Form1.Image1.Width := Round(Form1.Image1.Picture.Width*scale);
// так как Strech = True и размер области пропорционален
// размеру картинки, то картинка масштабируется без искажений
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
 If N_Image = 4 Then N_Image:=1;
 // Формирование имени картинки
 AFile:=IntToStr(N_Image)+ '.bmp';
 //Установка значения свойства Picture для вывода картинки
 // во время работы программы
 Form1.Image1.Picture.LoadFromFile(AFile);
 // Масштабирование картинки
 ScaleImage;
 N_Image:=N_Image+1;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 Image1.AutoSize := False;
 Image1.Stretch := True;     // Разрешим масштабирование

   // Запомним первоначальный размер области вывода иллюстрации
   iw := Image1.Width;
   ih := image1.Height;

   N_Image:=1;  // Начальное значение номера выводимой картинки
end;
end.
```

По материалам книги Никиты Культина «Основы программирования в Delphi 7»

Шестой этап. Заключительный

Самостоятельно внесите изменения в программу, добавив вывод картинок в зависимости от количества допущенных ошибок.

Протестируйте полученную программу.

Программа имеет еще один недостаток - после окончания игры (вывод панели с сообщением об окончании игры), если нажимать кнопки с буквами, то программа может аварийно завершиться. Чтобы этого не было в начале процедуры Poisk выполните проверку:

If (N_Attempt<Attempt) and (STR_N<>STR_R) then

Т.е. выполнять поиск нажатой буквы только в случае, если не исчерпаны все попытки и не отгадано все слово.