

Министерство образования и науки
Российской Федерации
Московский государственный университет
геодезии и картографии

ВВЕДЕНИЕ В DELPHI



Москва
2014

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ ГЕОДЕЗИИ И КАРТОГРАФИИ

И.И. Лонский, П.Д. Кужелев, А.С. Матвеев

Введение в Delphi

Москва
2014

Рецензенты:

профессор кафедры прикладной информатики МИИГАиК

А.П. Галеев;

профессор, доктор техн. наук **В.Н. Баранов**

(Государственный университет по землеустройству)

Составители: **И.И. Лонский, П.Д.Кужелев, А.С.Матвеев**

Введение в Delphi: Методические рекомендации.

МИИГАиК. – М.: 2014. – 46 с.

Методические рекомендации подробно рассказывают как создать в среде Delphi на языке Pascal свою первую программу. Составлены на основе учебной литературы и сведений из сети Интернет.

Для студентов V курса факультета прикладной космонавтики по предмету «Информатика».

Электронная версия методических рекомендаций размещена на сайте библиотеки МИИГАиК <http://library.miigaik.ru>

Введение

Delphi — это греческий город, знаменитый своим святилищем Аполлона в Фокиде. (Дельфийский оракул). Дельфийский оракул - общегреческий религиозный центр у подножья Парнаса. Этим именем фирма Borland назвала программный продукт с удивительными возможностями. Delphi — это комбинация нескольких важнейших технологий (высокопроизводительный компилятор в машинный код, объектно-ориентированная модель компонент, визуальное (а, следовательно, и скоростное) построение приложений из программных прототипов и др.). Delphi обеспечивает быструю разработку программ. В процессе построения приложения разработчик выбирает из палитры компонент готовые компоненты как художник, делающий крупные мазки кистью. Еще до компиляции он видит результаты своей работы. Среда Delphi включает в себя полный набор визуальных инструментов для скоростной разработки приложений (RAD — rapid application development), поддерживающей разработку пользовательского интерфейса. Библиотека визуальных компонент, включает в себя стандартные объекты построения пользовательского интерфейса, объекты управления данными, графические объекты, объекты мультимедиа, диалоги и объекты управления файлами.

Two-way tools — однозначное соответствие между визуальным проектированием и классическим написанием текста программы. Это означает, что разработчик всегда может видеть код, соответствующий тому, что он построил при помощи визуальных инструментов и наоборот.

Визуальный построитель интерфейсов (Visual User-interface builder) дает возможность быстро создавать приложения визуально, просто выбирая компоненты из соответствующей палитры. Структурное объектно-ориентированное программирование Delphi использует структурный объектно-ориентированный язык (Object Pascal), который сочетает с одной стороны выразительную мощь и простоту программирования. После запуска Delphi в верхнем окне горизонтально располагаются иконки палитры компонент. Если курсор задерживается на одной из иконок, под ней в желтом прямоугольнике появляется подсказка.

Из этой палитры компонент вы можете выбирать компоненты, из которых можно строить приложения. Компоненты включают в себя как визуальные, так и логические компоненты. Такие вещи, как кнопки, поля редактирования — это визуальные компоненты; а таблицы, отчеты — это логические.

Понятно, что поскольку в Delphi вы визуальным образом строите свою программу, все эти компоненты имеют свое графическое представление в поле форм для того, чтобы можно было бы ими соответствующим образом оперировать. Но для работающей программы видимыми остаются только визуальные компоненты. Компоненты сгруппированы на страницах палитры по своим функциям. К примеру, компоненты, представляющие Windows "common dialogs" все размещены на странице палитры с названием "Dialogs".

Инспектор объектов этот инструмент представляет из себя отдельное окно, где вы можете в период

проектирования программы устанавливать значения свойств и событий объектов (Properties & Events).

Формы, модули и метод разработки "Two-Way Tools"

Формы — это объекты, в которые вы помещаете другие объекты для создания пользовательского интерфейса вашего приложения. Модули состоят из кода, который реализует функционирование вашего приложения, обработчики событий для форм и их компонент.

Информация о формах хранится в двух типах файлов — .dfm и .pas, причем первый тип файла — двоичный — хранит образ формы и ее свойства, второй тип описывает функционирование обработчиков событий и поведение компонент. Оба файла автоматически синхронизируются

Delphi, так что если добавить новую форму в ваш проект, связанный с ним файл .pas автоматически будет создан, и его имя будет добавлено в проект.

Такая синхронизация и делает Delphi two-way-инструментом, обеспечивая полное соответствие между кодом и визуальным представлением. Как только вы добавите новый объект или код, Delphi устанавливает т.н. "кодovou синхронизацию" между визуальными элементами и соответствующими им кодовыми представлениями.

Например, предположим, вы добавили описание поведения формы (соотв. Обработчик событий), чтобы показывать окно сообщения по нажатию кнопки. Такое описание появляется, если дважды щелкнуть мышкой непосредственно на объект Button в форме или дважды щелкнуть мышью на строчку

OnClick на странице Events в Инспекторе объектов. В любом случае Delphi создаст процедуру или заголовок метода, куда вы можете добавить код.

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
end;
```

Создавая этот код, Delphi автоматически формирует декларацию объекта TForm1, которая содержит процедуру ButtonClick, представляющую из себя собственно обработчик события.

```
TForm1 = class (TForm)  
  Button1: Tbutton;  
  procedure Button1Click(Sender: TObject);  
private  
  { Private declarations }  
public  
  { Public declarations }  
end;
```

Конечно, вы запросто можете решить после получения этого кода, что автоматически созданные имена Вас не устраивают, и заменить их. Например, Button1 на Warning. Это можно сделать, изменив свойство Name для Button1 при помощи Инспектора объектов. Как только вы нажмете Enter, Delphi автоматически произведет соответствующую синхронизацию в коде.

Так как объект TForm1 существует в коде, вы свободно можете добавлять любые другие поля, процедуры, функции или

object definition. К примеру, вы можете дописать в коде свою собственную процедуру, обрабатывающую событие, а не делать это визуальным методом.

Следующий пример показывает, как это можно сделать. Обработчик принимает аргумент типа TObject, который позволяет нам определить, если необходимо, кто инициировал событие. Это полезно в случае, когда несколько кнопок вызывают общую процедуру для обработки.

```
TForm1 = class(TForm)
  Warning: TButton;
  Button1: TButton;
  procedure WarningClick(Sender: TObject);
  procedure NewHandler(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
```

Здесь мы имеем дело уже с второй стороной синхронизации. Визуальная среда в данном случае распознает, что новая процедура добавлена к объекту и соответствующие имена появляются в Инспекторе объектов.

Прежде всего, для того чтобы научиться программировать на Delphi надо знать основы языка Pascal. Предположим, вы уже знакомы с Pascal'ем. Запустите Delphi. Перед Вами появится окошко с заголовком Form1 — оно показано на рисунке красной

стрелкой. Это ваша рабочая форма на нее вы будете помещать компоненты для работы.

Компонент — это объект, который позволяет положить его на форму для работы с ним, для примера кнопка — это компонент и т.д, компоненты можно поставить при помощи палитры компонентов...

Палитра компонентов — это строка с кнопками на ней, находящаяся обычно в правом верхнем углу экрана она показана стрелкой на втором рисунке. Чтобы установить компонент, надо кликнуть на кнопку нужного вам компонента, а затем кликнуть на форму и компонент установится, вы так-же сможете передвигать и изменять размеры компонента мышкой за его границы. Компоненты бывают видимые и невидимые: видимые — компоненты которые видно на откомпилированной программе это такие компоненты как кнопки (объект TButton), списки (TListBox), мемо редакторы (TMemo) и т.д. Невидимые компоненты, которые не видно в финальной программе и которые представляют собой объекты оформленные как компоненты, для удобства их использования, ведь как известно компоненты на данный момент являются самым высшим уровнем программирования, а значит и самым удобным и простым.

Для того чтобы настраивать свойства компонентов используется Object Inspector (рис. 1–3).

Object Inspector — это окошко, обозначенное стрелкой на рис. 3, в нем вы можете поставить все свойства объектов. Например, на рисунке в Object Inspector`е находятся свойства объекта TForm который представляет из себя рабочую форму: если найти там свойство Caption и

изменить его с "Form1" к примеру, на "My Form" то заголовок окна изменится на введенный. Выберите на палитре компонентов компонент Button(TButton) так как показано на рис. 1. Поставьте его на форму как на рис. 2. Нажмите F9 и программа откомпилируется и запустится...

Получилась одна из самых простейших программ написанных на Delphi, а теперь усложним задачу - предположим нам надо сделать, чтобы по нажатию на кнопку высвечивалось сообщение об ошибке или просто сообщение с текстом: "Привет". Для этого нам надо будет обработать (т.е. определить) когда кнопку нажали.

В Windows все действия, такие как нажатие на кнопку и т. п. происходят путем сообщений:

Сообщение — это структура определенного типа, которая передает информацию от одного объекта к другому . Но не будем вдаваться в WIN API и займемся программой...

В Delphi объекты сами умеют обрабатывать некоторые сообщения - так и кнопка умеет обрабатывать нажатие на нее. Для того чтобы задать процедуру реакции на нажатие кнопки надо: переключиться в Object Inspector`е на Events как показано на рис. 3, найти надпись — OnClick (показано красной стрелкой — рис. 3) и выбрать имя процедуры в выплывающем списке показанным синей стрелочкой, если вы еще не сделали процедуру, то просто щелкните 2 раза на список и процедуру напишет сам Delphi — а в процедуре уже хозяин вы. Но все же надо ведь сделать вывод сообщения: `procedure ShowMessage(const msg:String)` выводит на экран окно с сообщением...

Итак финальный текст модуля:

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics,
  Controls, Forms, Dialogs,
  StdCtrls;
type
  TForm1 = class(TForm)
  Button1: TButton;
  procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
  {$R *.DFM}
  procedure TForm1.Button1Click(Sender: TObject);
  begin
    showmessage('Привет'); // - единственная написанная
строка модуля
  end;
end.
```

Первая программа

В качестве примера напишем программу игры Tic-tac-toe (Крестики-нолики).

Конечно, на разработку серьезной информационно-поисковой системы в архитектуре клиент-сервер может уйти гораздо большее время, чем на разработку программы-игрушки. Тем не менее, *начинать надо с простого.*

Данное методическое пособие не ставит перед собой цель дублировать справку Delphi либо описывать подробно интерфейс системы разработки.

Задача данного методического пособия поставить цель, которую, Вы можете достичь без специальных знаний и помочь Вам на этом пути. Предполагается что все термины и вопросы, упоминаемые, но не рассматриваемы ниже — будут проработаны Вами самостоятельно.

Тема I. Создаем приложение

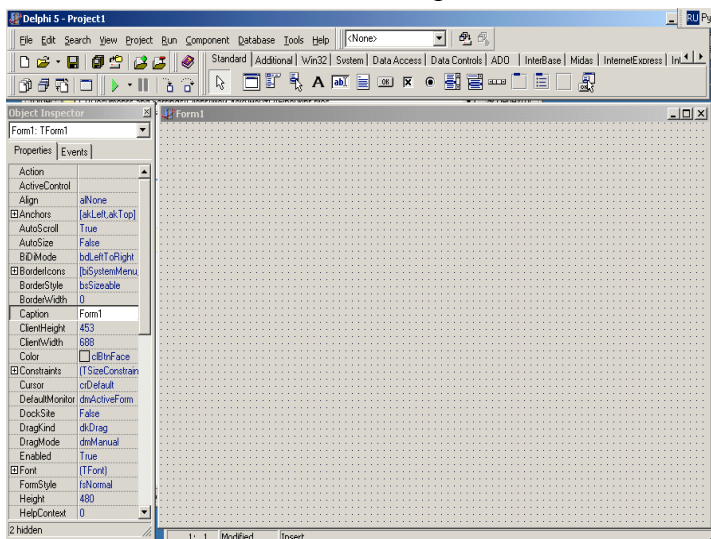


Рис.1

Запустите среду программирования Delphi. В главном

меню выберите пункт File/New Application. После его выполнения Вы увидите созданные окна :

Окно визуальной представления Вашей программы (рис. 1) и Окно кода (рис. 2) .

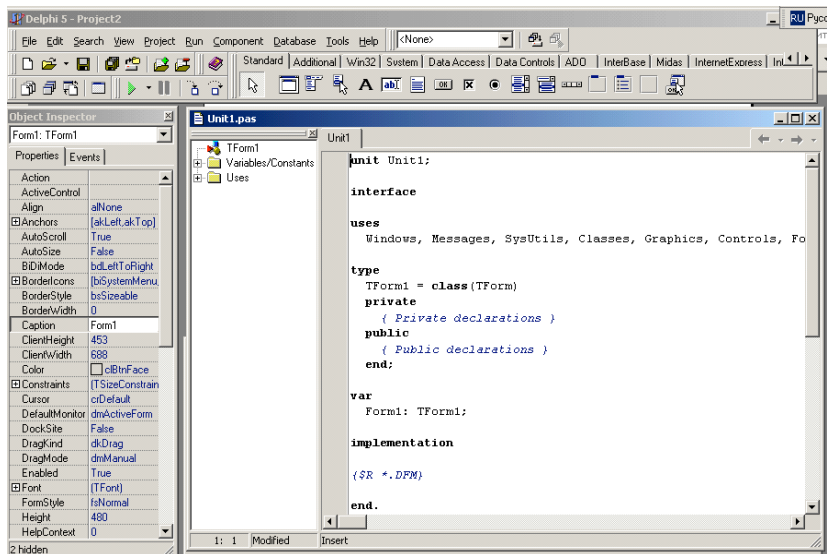


Рис.2.

В окне кода программы Вы увидите следующий текст:

```
unit Unit1;  
interface  
uses  
  Windows, Messages, SysUtils, Classes, Graphics, Controls,  
  Forms, Dialogs;  
type  
  TForm1 = class(TForm)  
  private
```

```
{ Private declarations }  
public  
{ Public declarations }  
end;  
var  
    Form1: TForm1;  
implementation  
{ $R *.DFM }  
end.  
Листинг 1
```

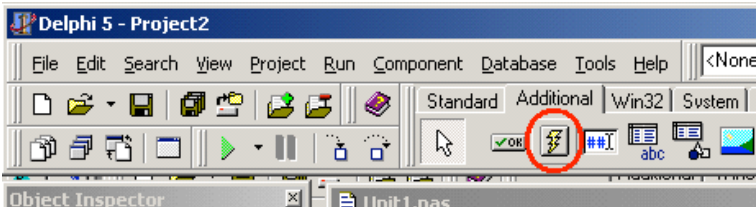


Рис.3.

Выберем в панели компонент закладку Additional.

Нас интересует компонент отмеченный кружком на рис. 3 – TSpeedButton

Поместим на форму 9 экземпляров кнопки (SpeedButton) в соответствии с рис. 4.

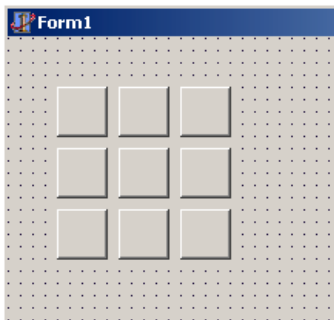


Рис.4.

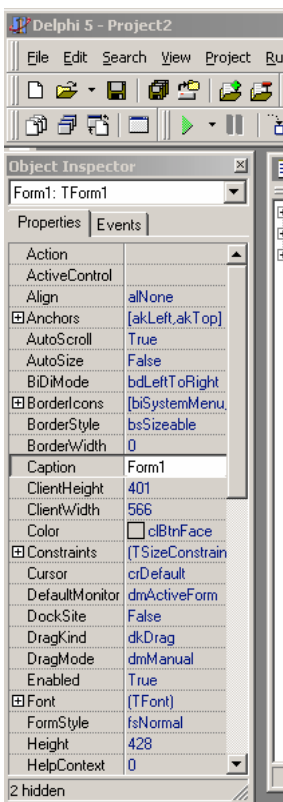


Рис. 5

Обратим внимание на окно Object Inspector (рис. 5)

Выберем закладку Events в Object inspector (рис. 6)

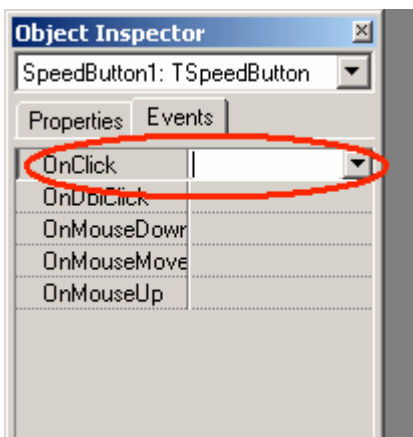


Рис. 6

Кликнем мышкой справа от события OnClick

В окне коды автоматически будет создан следующий текст:

```
procedure TForm1.SpeedButton1Click(Sender: TObject);  
begin  
  
end;
```

Листинг 2

Выберем любую созданную нами кнопку. В Object Inspector найдем

Свойство кнопки — Caption. Написанный там текст будет выводиться на самой кнопке. Попробуйте. Теперь сотрите текст. Попробуем сделать то же самое программно.

В процедуре обрабатывающей нажатие на кнопку мышки над данной кнопкой (Листинг 2.) напишем (Событие-Event):

```
SpeedButton1.Caption:=’Тест’;
```

Наша первая программа готова! Для запуска нашей программы нажимаем F9 на клавиатуре. Проверяем работоспособность программы.

Если Вы все сделали правильно, то при нажатии на выбранную кнопку (только для той кнопки, для которой Вы создали, написали обработчик нажатия) на ней появится надпись “Тест”.

Так как мы пишем игру ‘крестики нолики’ — то в каждой ячейке игрового поля 3x3 мы будем ставить “X” либо “0”.

Попробуйте написать обработчики нажатия для всех кнопок — выводя при нажатии крестики.

Состояние каждого поля игры может принимать три значения:

- Пусто
- Крестик
- Нолик.

Договоримся что первый игрок будет ходить крестиками. Создадим переменную, в которой будем хранить тип предстоящего хода – крестик или нолик. Для

этого в секции описания переменных (Var) напишем:

```
Xod:char='X';
```

А в обработчике нажатия на кнопку напишем:

```
SpeedButton1.Caption:=Xod;
```

Программа будет выглядеть следующим образом:

```
unit Unit1;  
interface  
uses  
  Windows, Messages, SysUtils, Classes, Graphics,  
Controls, Forms, Dialogs,  
  Buttons;  
type  
  TForm1 = class(TForm)  
    SpeedButton1: TSpeedButton;  
    SpeedButton2: TSpeedButton;  
    SpeedButton3: TSpeedButton;  
    SpeedButton4: TSpeedButton;  
    SpeedButton5: TSpeedButton;  
    SpeedButton6: TSpeedButton;  
    SpeedButton7: TSpeedButton;  
    SpeedButton8: TSpeedButton;  
    SpeedButton9: TSpeedButton;  
    procedure SpeedButton1Click(Sender: TObject);  
private  
  { Private declarations }  
public  
  { Public declarations }
```

```
end;  
var  
  Form1: TForm1;  
  Xod:char='X';  
implementation  
{$R *.DFM}  
procedure TForm1.SpeedButton1Click(Sender: TObject);  
begin  
  SpeedButton1.Caption:='Тест';  
end;  
end.
```

Листинг 3

Запустим программу и протестируем.

При нажатии на те клавиши, для которых Вы написали обработчики (в Листинге 3. только для одной) в соответствующие кнопки будет выводиться крестик.

Теперь нам надо написать код который позволит следующему игроку ставить нолик.

Нам необходимо учесть следующее:

1.Если на кнопке нет никакого символа – то текущий знак ставить можно, а если есть - то нельзя.

2.После присвоения кнопки знака текущий знак должен меняться на противоположный.

Напишем в каждом обработчике следующий код (пример только для одной кнопки):

```

procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
if SpeedButton1.Caption='' then
begin
SpeedButton1.Caption:=Xod;
if Xod='X' then Xod='0'
else
Xod:='0';
end;
end;

```

Листинг 4

Протестируем программу.

Напишем обработчики для всех кнопок.

```

unit Unit1;
interface
uses
Windows, Messages, SysUtils, Classes, Graphics,
Controls, Forms, Dialogs,
Buttons;
type
TForm1 = class(TForm)
SpeedButton1: TSpeedButton;
SpeedButton2: TSpeedButton;
SpeedButton3: TSpeedButton;
SpeedButton4: TSpeedButton;
SpeedButton5: TSpeedButton;

```

```

SpeedButton6: TSpeedButton;
SpeedButton7: TSpeedButton;
SpeedButton8: TSpeedButton;
SpeedButton9: TSpeedButton;
procedure SpeedButton1Click(Sender: TObject);
procedure SpeedButton2Click(Sender: TObject);
procedure SpeedButton3Click(Sender: TObject);
procedure SpeedButton4Click(Sender: TObject);
procedure SpeedButton5Click(Sender: TObject);
procedure SpeedButton6Click(Sender: TObject);
procedure SpeedButton7Click(Sender: TObject);
procedure SpeedButton8Click(Sender: TObject);
procedure SpeedButton9Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
var
  Form1: TForm1;
  Xod:char='X';
implementation
  {$R *.DFM}
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
if SpeedButton1.Caption='' then
  begin
    SpeedButton1.Caption:=Xod;
  end
end;

```

```

    if Xod='X' then Xod:='0'
      else
        Xod:='X';
      end;
end;
procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
  if SpeedButton2.Caption='' then
    begin
      SpeedButton2.Caption:=Xod;
      if Xod='X' then Xod:='0'
        else
          Xod:='X';
        end;
    end;
end;
procedure TForm1.SpeedButton3Click(Sender: TObject);
begin
  if SpeedButton3.Caption='' then
    begin
      SpeedButton3.Caption:=Xod;
      if Xod='X' then Xod:='0'
        else
          Xod:='X';
        end;
    end;
end;
procedure TForm1.SpeedButton4Click(Sender: TObject);
begin
  if SpeedButton4.Caption='' then

```

```

begin
  SpeedButton4.Caption:=Xod;
  if Xod='X' then Xod:='0'
    else
      Xod:='X';
  end;
end;
procedure TForm1.SpeedButton5Click(Sender: TObject);
begin
  if SpeedButton5.Caption='' then
  begin
    SpeedButton5.Caption:=Xod;
    if Xod='X' then Xod:='0'
      else
        Xod:='X';
    end;
  end;
procedure TForm1.SpeedButton6Click(Sender: TObject);
begin
  if SpeedButton6.Caption='' then
  begin
    SpeedButton6.Caption:=Xod;
    if Xod='X' then Xod:='0'
      else
        Xod:='X';
    end;
  end;
procedure TForm1.SpeedButton7Click(Sender: TObject);

```

```

begin
if SpeedButton7.Caption='' then
begin
    SpeedButton7.Caption:=Xod;
    if Xod='X' then Xod:='0'
        else
            Xod:='X';
        end;
end;
procedure TForm1.SpeedButton8Click(Sender: TObject);
begin
if SpeedButton8.Caption='' then
begin
    SpeedButton8.Caption:=Xod;
    if Xod='X' then Xod:='0'
        else
            Xod:='X';
        end;
end;
procedure TForm1.SpeedButton9Click(Sender: TObject);
begin
if SpeedButton9.Caption='' then
begin
    SpeedButton9.Caption:=Xod;
    if Xod='X' then Xod:='0'
        else
            Xod:='Ö';
        end;
end;

```

end;
end.

Листинг 5

Обратите внимание на отступы в тексте программы. Так код читается легче.

Протестируйте программу. Если Вы все сделали правильно, то можете сыграть сами с собой.

Теперь приведем нашу программу к более красивому виду.

У формы, на которой Вы поместили кнопки, то же есть свойства. И то же есть свойство Caption. Напишем в соответствующей этому свойству ячейке следующий текст: Крестики-нолики

Сделаем размеры нашей формы поменьше. А в свойстве формы

BorderStyle укажем bsDialog.

Запустим программу.

Она должна выглядеть так (рис. 7):

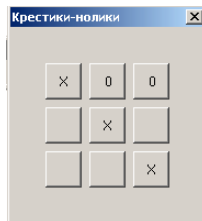


Рис.7.

Теперь нам надо сделать процедуру проверки окончания игры.

Будем считать игру законченной, если три крестика, либо нолика стоят подряд в любом направлении либо всем кнопкам присвоены символы (ничья).

```
Procedure CheckGame;  
begin  
  with Form1 do  
  begin  
    if ((SpeedButton1.Caption=SpeedButton2.Caption)  
      and  
      (SpeedButton1.Caption<>'')  
    (SpeedButton2.Caption=SpeedButton3.Caption))  
      or  
      ((SpeedButton4.Caption=SpeedButton5.Caption)  
        and  
        (SpeedButton4.Caption<>'')  
      and  
      (SpeedButton5.Caption=SpeedButton6.Caption))  
        or  
        ((SpeedButton7.Caption=SpeedButton8.Caption)  
          and  
          (SpeedButton7.Caption<>'')  
        and  
        (SpeedButton8.Caption=SpeedButton9.Caption))  
          or  
          ((SpeedButton1.Caption=SpeedButton4.Caption)  
            and  
            (SpeedButton1.Caption<>'')
```

```
and
(SpeedButton4.Caption=SpeedButton7.Caption))
or
((SpeedButton2.Caption=SpeedButton5.Caption)
and
(SpeedButton2.Caption<>"")
and
(SpeedButton5.Caption=SpeedButton8.Caption))
or
((SpeedButton3.Caption=SpeedButton6.Caption)
and
(SpeedButton3.Caption<>"")
and
(SpeedButton6.Caption=SpeedButton9.Caption))
or
((SpeedButton1.Caption=SpeedButton5.Caption)
and
(SpeedButton1.Caption<>"")
and
(SpeedButton5.Caption=SpeedButton9.Caption))
or
((SpeedButton3.Caption=SpeedButton5.Caption)
and
(SpeedButton3.Caption<>"")
and
(SpeedButton5.Caption=SpeedButton7.Caption))
then
    ShowMessage('Игра окончена!');
```

end;

Листинг 5

Будем вызывать процедуру `CheckGame` в каждом обработчике нажатия.

Вот так:

Листинг 6

```
procedure TForm1.SpeedButton1Click(Sender: TObject);  
begin  
if SpeedButton1.Caption="" then  
    begin  
        SpeedButton1.Caption:=Xod;  
        if Xod='X' then Xod:='0'  
            else  
                Xod:='X';  
            end;  
        CheckGame;  
    end;  
end;
```

Для проверки на “ничью” напишем процедуру

CheckGame2.

```
function CheckGame2:boolean;  
begin  
    Result:=false;  
    with Form1 do  
        begin  
            if (SpeedButton1.Caption<>"")  
                and  
                (SpeedButton2.Caption<>"")  
                and  
                (SpeedButton3.Caption<>"")  
                and  
                (SpeedButton4.Caption<>"")  
                and  
                (SpeedButton5.Caption<>"")  
                and  
                (SpeedButton6.Caption<>"")  
                and  
                (SpeedButton7.Caption<>"")  
                and  
                (SpeedButton8.Caption<>"")  
                and  
                (SpeedButton9.Caption<>"")  
            then  
                Result:=true;  
            end;  
        end;
```

Листинг 7

При проверке на “ничью” либо на выигрыш одного из участников уместно задать ему вопрос, что делать далее: закончить игру либо сыграть еще раз.

Напишем данный диалог в процедуре CheckGame.

```
if MessageDlg('Закончить игру?',  
mtConfirmation,[mbYes,mbNo],0)=mrYes  
then  
ClearGame  
else  
Halt;
```

Листинг 8

Процедура ClearGame стирает все знаки с кнопок.

Вот ее код.

```
Procedure ClearGame;  
begin  
with Form1 do  
begin  
SpeedButton1.Caption:='';  
SpeedButton2.Caption:='';  
SpeedButton3.Caption:='';  
SpeedButton4.Caption:='';  
SpeedButton5.Caption:='';  
SpeedButton6.Caption:='';  
SpeedButton7.Caption:='';  
SpeedButton8.Caption:='';  
SpeedButton9.Caption:='';
```

```
end;  
end;
```

Листинг 9

И в итоге наша программа выглядит следующим образом:

```
unit Unit1;  
interface  
uses  
  Windows, Messages, SysUtils, Classes, Graphics,  
  Controls, Forms, Dialogs,  
  Buttons;  
type  
  TForm1 = class(TForm)  
    SpeedButton1: TSpeedButton;  
    SpeedButton2: TSpeedButton;  
    SpeedButton3: TSpeedButton;  
    SpeedButton4: TSpeedButton;  
    SpeedButton5: TSpeedButton;  
    SpeedButton6: TSpeedButton;  
    SpeedButton7: TSpeedButton;  
    SpeedButton8: TSpeedButton;  
    SpeedButton9: TSpeedButton;  
  procedure SpeedButton1Click(Sender: TObject);  
  procedure SpeedButton2Click(Sender: TObject);  
  procedure SpeedButton3Click(Sender: TObject);  
  procedure SpeedButton4Click(Sender: TObject);
```

```

procedure SpeedButton5Click(Sender: TObject);
procedure SpeedButton6Click(Sender: TObject);
procedure SpeedButton7Click(Sender: TObject);
procedure SpeedButton8Click(Sender: TObject);
procedure SpeedButton9Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;
var
    Form1: TForm1;
    Xod:char='X';
implementation
{SR *.DFM}
Procedure ClearGame;
begin
    with Form1 do
    begin
    SpeedButton1.Caption:='';
        SpeedButton2.Caption:='';
        SpeedButton3.Caption:='';
        SpeedButton4.Caption:='';
        SpeedButton5.Caption:='';
        SpeedButton6.Caption:='';
        SpeedButton7.Caption:='';
        SpeedButton8.Caption:='';
        SpeedButton9.Caption:='';
    
```

```
    end;  
end;  
  
function CheckGame2:boolean;  
begin  
    Result:=false;  
    with Form1 do  
        begin  
            if (SpeedButton1.Caption<>'')  
                and  
                (SpeedButton2.Caption<>'')  
                and  
                (SpeedButton3.Caption<>'')  
                and  
                (SpeedButton4.Caption<>'')  
                and  
                (SpeedButton5.Caption<>'')  
                and  
                (SpeedButton6.Caption<>'')  
                and  
                (SpeedButton7.Caption<>'')  
                and  
                (SpeedButton8.Caption<>'')  
                and  
                (SpeedButton9.Caption<>'')  
            then  
                Result:=true;  
            end;  
        end;  
    end;  
end;
```

```

end;
Procedure CheckGame;
begin
  if CheckGame2 then
    begin
      if
        MessageDlg('Игра
играть?',mtConfirmation,[mbYes,mbNo],0)=mrYes
      then
        begin
          ClearGame;
          exit;
        end
      else
        Halt;
    end;
  with Form1 do
    begin

      if ((SpeedButton1.Caption=SpeedButton2.Caption)
        and
        (SpeedButton1.Caption<>''))
        and
        (SpeedButton2.Caption=SpeedButton3.Caption))
        or
        ((SpeedButton4.Caption=SpeedButton5.Caption)
        and
        (SpeedButton4.Caption<>''))
        and

```

**(SpeedButton5.Caption=SpeedButton6.Caption))
or
((SpeedButton7.Caption=SpeedButton8.Caption)
and
(SpeedButton7.Caption<>'')
and
(SpeedButton8.Caption=SpeedButton9.Caption))
or
((SpeedButton1.Caption=SpeedButton4.Caption)
and
(SpeedButton1.Caption<>'')
and
(SpeedButton4.Caption=SpeedButton7.Caption))
or
((SpeedButton2.Caption=SpeedButton5.Caption)
and
(SpeedButton2.Caption<>'')
and
(SpeedButton5.Caption=SpeedButton8.Caption))
or
((SpeedButton3.Caption=SpeedButton6.Caption)
and
(SpeedButton3.Caption<>'')
and
(SpeedButton6.Caption=SpeedButton9.Caption))
or
((SpeedButton1.Caption=SpeedButton5.Caption)
and**

```

(SpeedButton1.Caption<>'')
and
(SpeedButton5.Caption=SpeedButton9.Caption))
or
((SpeedButton3.Caption=SpeedButton5.Caption)
and
(SpeedButton3.Caption<>'')
and
(SpeedButton5.Caption=SpeedButton7.Caption))
then
if      MessageDlg('Игра      закончена!
еще?',mtConfirmation,[mbYes,mbNo],0)=mrYes
then
ClearGame
else
Halt;
end;
end;
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
if SpeedButton1.Caption='' then
begin
SpeedButton1.Caption:=Xod;
if Xod='X' then Xod:='0'
else
Xod:='X';
end;
CheckGame;

```

```
end;  
procedure TForm1.SpeedButton2Click(Sender: TObject);  
begin  
if SpeedButton2.Caption='' then  
begin  
SpeedButton2.Caption:=Xod;  
if Xod='X' then Xod:='0'  
else  
Xod:='X';  
end;  
CheckGame;  
end;
```

```
procedure TForm1.SpeedButton3Click(Sender: TObject);  
begin  
if SpeedButton3.Caption='' then  
begin  
SpeedButton3.Caption:=Xod;  
if Xod='X' then Xod:='0'  
else  
Xod:='X';  
end;  
CheckGame;  
end;
```

```
procedure TForm1.SpeedButton4Click(Sender: TObject);  
begin  
if SpeedButton4.Caption='' then
```

```
begin
  SpeedButton4.Caption:=Xod;
  if Xod='X' then Xod:='0'
    else
      Xod:='X';
    end;
  CheckGame;
end;
```

```
procedure TForm1.SpeedButton5Click(Sender: TObject);
begin
  if SpeedButton5.Caption='' then
  begin
    SpeedButton5.Caption:=Xod;
    if Xod='X' then Xod:='0'
      else
        Xod:='X';
      end;
    CheckGame;
  end;
```

```
procedure TForm1.SpeedButton6Click(Sender: TObject);
begin
  if SpeedButton6.Caption='' then
  begin
    SpeedButton6.Caption:=Xod;
    if Xod='X' then Xod:='0'
      else
```

```
    Xod:='X';  
    end;  
    CheckGame;  
end;
```

```
procedure TForm1.SpeedButton7Click(Sender: TObject);  
begin  
if SpeedButton7.Caption='' then  
begin  
    SpeedButton7.Caption:=Xod;  
    if Xod='X' then Xod:='0'  
    else  
        Xod:='X';  
    end;  
    CheckGame;  
end;  
end;
```

```
procedure TForm1.SpeedButton8Click(Sender: TObject);  
begin  
if SpeedButton8.Caption='' then  
begin  
    SpeedButton8.Caption:=Xod;  
    if Xod='X' then Xod:='0'  
    else  
        Xod:='X';  
    end;  
  
    CheckGame;
```

end;

procedure TForm1.SpeedButton9Click(Sender: TObject);

begin

if SpeedButton9.Caption='' then

begin

SpeedButton9.Caption:=Xod;

if Xod='X' then Xod:='0'

else

Xod:='Õ';

end;

CheckGame;

end;

end.

Листинг 10.

Вернем еще раз к внешнему виду нашей программы.

У каждой кнопки есть свойство Cursor – по умолчанию оно выставлено в

Cr Default. Заменяем на crHandPoint.

У формы есть свойство Color . Заменяем на clNavy.

Создадим еще одну кнопку и напишем на ней Выход.

При нажатии на эту кнопку программа будет закрыта.

В главном меню выберите пункт File/Save project As ...

И укажите для проекта имя game. Выберите директорию для программы и сохраните под новым

именем “game”.

Поздравляем Вас с первой самостоятельно написанной программой!

Теперь усложним задачу.

Игра с компьютером

Выберем компонент RadioGroup (закладка Standard) и поместим его на форму. Установим для него свойство цвет шрифта (Font) в желтый цвет. А в свойстве Items добавим два пункта

- Человек
- Компьютер

Свойство Caption изменим на “Противник”.

Давайте подумаем, как в варианте противник-компьютер будет делаться ответный ход. Вариантов много. Начнем с самого простого варианта – Случайный выбор.

Напишем процедуру Computer.

```
Procedure Computer;  
begin  
  with Form1 do  
    begin  
      if Radiogroup1.ItemIndex=0 then exit;  
      if (SpeedButton1.Caption='') then  
        begin  
          SpeedButton1.Caption:=Xod;  
          if Xod='X' then Xod:='0'  
        end  
      end  
    end  
end
```

```
    else
      Xod:='X';
      CheckGame;
      exit;
end;
if (SpeedButton2.Caption='') then
begin
  SpeedButton2.Caption:=Xod;
  if Xod='X' then Xod:='0'
  else
    Xod:='X';
    CheckGame;
    exit;
end;
if (SpeedButton3.Caption='') then
begin
  SpeedButton3.Caption:=Xod;
  if Xod='X' then Xod:='0'
  else
    Xod:='X';
    CheckGame;
    exit;
end;
if (SpeedButton4.Caption='') then
begin
  SpeedButton4.Caption:=Xod;
  if Xod='X' then Xod:='0'
  else
```

```
Xod:='X';
CheckGame;
exit;
end;
if (SpeedButton5.Caption='') then
begin
    SpeedButton5.Caption:=Xod;
    if Xod='X' then Xod:='0'
    else
        Xod:='X';
    CheckGame;
    exit;
end;
if (SpeedButton6.Caption='') then
begin
    SpeedButton6.Caption:=Xod;
    if Xod='X' then Xod:='0'
    else
        Xod:='X';
    CheckGame;
    exit;
end;
if (SpeedButton7.Caption='') then
begin
    SpeedButton7.Caption:=Xod;
    if Xod='X' then Xod:='0'
    else
        Xod:='X';
```

```

    CheckGame;
    exit;
end;
if (SpeedButton8.Caption='') then
begin
    SpeedButton8.Caption:=Xod;
    if Xod='X' then Xod:='0'
    else
    Xod:='X';
    CheckGame;
    exit;
end;
if (SpeedButton9.Caption='') then
begin
    SpeedButton9.Caption:=Xod;
    if Xod='X' then Xod:='0'
    else
    Xod:='X';
    CheckGame;
    exit;
end;
end;
end;

```

Листинг 11.

Процедуру CheckGame объявим до секции IMPLEMENTATION

А в каждой процедуре – обработчике в строке перед последним end – напомним название процедуры “Computer”.

Получим программу играющую в двух режимах
(рис. 8):

“Человек-человек” И “Человек-компьютер”

Надо отметить, что весь вышеприведенный код можно записать гораздо компактней, но авторы для простоты изложения предпочли не усложнять программу.

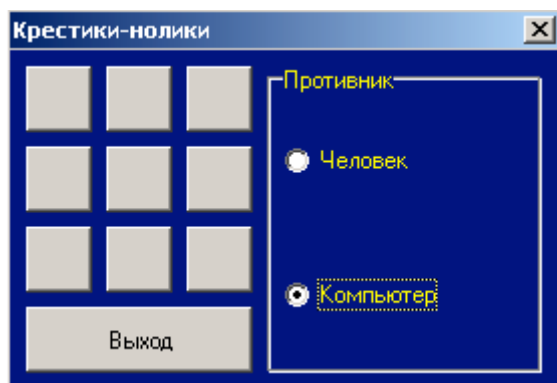


Рис.8

Примерная тематика самостоятельных работ по
программированию:

1. Перевести программу на графическое отображение игрового поля.
2. Сделать несколько уровней сложности при игре с компьютером и далее...
3. Игра шашки.
4. Игра Го.
5. Игра морской бой.

Внутривузовское издание

Подписано в печать 15.12.2014. Гарнитура Таймс

Формат 60·90/16. Бумага офсетная.

Объем 3,0 усл. печ. л.

Тираж 15 экз. Заказ №196 Продаже не подлежит

Отпечатано в УПП «Репрография» МИИГАиК