```
unit Unit1;
interface
uses
 System.SysUtils, System.Classes, Vcl.Controls, Vcl.Forms, Vcl.Dialogs,
 Vcl.Grids, Vcl.StdCtrls, Vcl.Imaging.jpeg, Vcl.ExtCtrls, Windows, Graphics,
 Vcl.Buttons, Vcl.Menus, ShellApi;
type
 TForm1 = class(TForm)
 StringGrid1: TStringGrid;
 ComboBox1: TComboBox;
 Image1: TImage;
 SpeedButton1: TSpeedButton;
 SpeedButton2: TSpeedButton;
 SpeedButton3: TSpeedButton;
 SpeedButton4: TSpeedButton;
 MainMenu1: TMainMenu;
 N1: TMenuItem;
 Label1: TLabel;
 procedure FormCreate(Sender: TObject);
 procedure ComboBox1Change(Sender: TObject);
 procedure StringGrid1DrawCell(Sender: TObject; ACol, ARow: Integer; Rect: TRect; State: TGridDrawState);
 procedure StringGrid1SelectCell(Sender: TObject; ACol, ARow: Integer;
 var CanSelect: Boolean);
 procedure LoadLevelFromFile(const FileName: string);
 procedure SpeedButton1Click(Sender: TObject);
 procedure SpeedButton2Click(Sender: TObject);
 procedure SpeedButton3Click(Sender: TObject);
 procedure SpeedButton4Click(Sender: TObject);
 procedure N1Click(Sender: TObject);
 private
 CompletedLevels: array[0..4] of Boolean;
 MaxAvailableLevel: Integer;
 LevelsCompleted: Integer;
 Sudoku: array[0..8, 0..8] of Integer;
 Solution: array[0..8, 0..8] of Integer;
 procedure UpdateLevelsCompleted;
 procedure InitializeGrid;
 procedure LoadSudoku(Difficulty: Integer);
 function IsValid: Boolean;
 function SolveSudoku(var Grid: array of Integer): Boolean;
 function FindUnassignedLocation(var Grid: array of Integer; out Row, Col: Integer): Boolean;
 function IsSafe(var Grid: array of Integer; Row, Col, Num: Integer): Boolean;
 end;
var
 Form1: TForm1;
implementation
{$R *.dfm}
uses Unit3;
function Min(a, b: Integer): Integer;
begin
 if a < b then
 Result := a
 else
 Result := b;
end;
procedure TForm1.StringGrid1DrawCell(Sender: TObject; ACol, ARow: Integer; Rect: TRect; State: TGridDrawState);
var
 Grid: TStringGrid;
begin
 Grid := Sender as TStringGrid;
 // Устанавливаем цвет и ширину линии для границ
 Grid.Canvas.Pen.Color := clBlack;
 Grid.Canvas.Pen.Width := 2;
 // Рисуем вертикальные линии для внешних границ блоков 3x3
 if (ACol mod 3 = 0) and (ACol <> 0) then
 begin
 Grid.Canvas.MoveTo(Rect.Left, Rect.Top);
 Grid.Canvas.LineTo(Rect.Left, Rect.Bottom);
 end;
 // Рисуем горизонтальные линии для внешних границ блоков 3x3
 if (ARow mod 3 = 0) and (ARow <> 0) then
 begin
 Grid.Canvas.MoveTo(Rect.Left, Rect.Top);
 Grid.Canvas.LineTo(Rect.Right, Rect.Top);
 end;
 // Также рисуем правую и нижнюю внешние границы последнего блока, если это последний столбец или последняя строка
 if (ACol = Grid.ColCount - 1) and ((ACol + 1) mod 3 = 0) then
 begin
```

```
 Grid.Canvas.MoveTo(Rect.Right, Rect.Top);
 Grid.Canvas.LineTo(Rect.Right, Rect.Bot-
tom);
 end;
 if (ARow = Grid.RowCount - 1) and ((ARow
+ 1) mod 3 = 0) then
 begin
 Grid.Canvas.MoveTo(Rect.Left, Rect.Bot-
tom);
 Grid.Canvas.LineTo(Rect.Right, Rect.Bot-
tom);
 end;
 Grid.Canvas.TextRect(Rect, Rect.Left + 2,
Rect.Top + 2, Grid.Cells[ACol, ARow]);
end;
procedure TForm1.StringGrid1Se-
lectCell(Sender: TObject; ACol, ARow: Inte-
ger; var CanSelect: Boolean);
begin
 StringGrid1.Repaint;
end;
procedure TForm1.FormCreate(Sender:
TObject);
var
 i: Integer;
begin
 InitializeGrid;
 ComboBox1.Items.Add('Легкий');
 ComboBox1.Items.Add('Средний');
 ComboBox1.Items.Add('Сложный');
 ComboBox1.Items.Add('Эксперт');
 ComboBox1.Items.Add('Экстримальный');
 ComboBox1.ItemIndex := 0;
 ComboBox1.OnChange := Com-
boBox1Change; //Обработчик события
 LevelsCompleted := 0;
 Label1.Caption := 'Уровней пройдено: 0';
 MaxAvailableLevel := 0; // Начальное значе-
ние уровня (Легкий)
 for i := 0 to 4 do
 CompletedLevels[i] := False;
end;
procedure TForm1.LoadLevelFromFile(const
FileName: string);
var
 LevelFile: TextFile;
 Row, Col, Value: Integer;
begin
 AssignFile(LevelFile, FileName);
 Reset(LevelFile);
 // Очищаем StringGrid
 InitializeGrid;
 for Row := 0 to 8 do
 begin
 for Col := 0 to 8 do
 begin
 Read(LevelFile, Value);
 Sudoku[Row, Col] := Value;
 if Value = 0 then
 StringGrid1.Cells[Col, Row] := ''
 else
 StringGrid1.Cells[Col, Row] := IntTo-
Str(Value);
 end;
 end;
 CloseFile(LevelFile);
 // Принудительно перерисовываем String-
Grid
 StringGrid1.Invalidate;
end;
procedure TForm1.LoadSudoku(Difficulty: In-
teger);
var
 LevelFileName, FullPath: string;
begin
 // Очищаем сетку перед загрузкой новой
игры
 InitializeGrid;
 case Difficulty of
 0: LevelFileName := 'easy.txt'; // Легкий
 1: LevelFileName := 'medium.txt'; // Средний
 2: LevelFileName := 'hard.txt'; // Сложный
 3: LevelFileName := 'expert.txt'; // Эксперт
 4: LevelFileName := 'extreme.txt'; // Экстри-
мальный
 end;
 // Создаем полный путь к папке с уровнями
 FullPath := ExtractFilePath(Application.Ex-
eName) + 'Levels\' + LevelFileName;
 if FileExists(FullPath) then
 LoadLevelFromFile(FullPath)
 else
 ShowMessage('Файл уровня не найден: ' +
FullPath);
 SpeedButton3.Enabled := False; // Отключаем
кнопку "Solve"
end;
procedure TForm1.N1Click(Sender: TObject);
begin
 ShellExecute(0, 'open', PChar ('help.chm'), nil,
nil, SW_SHOW);
end;
```

```
procedure TForm1.InitializeGrid;
var
 i, j: Integer;
begin
 StringGrid1.RowCount := 9;
 StringGrid1.ColCount := 9;
 StringGrid1.Options := StringGrid1.Options +
[goEditing]; // Разрешить редактирование
ячеек
 for i := 0 to 8 do
 for j := 0 to 8 do
 begin
 StringGrid1.Cells[j, i] := '';
 Sudoku[i, j] := 0;
 Solution[i, j] := 0;
 end;
 // Принудительно перерисовываем
StringGrid
 StringGrid1.Invalidate;
end;
procedure TForm1.Com-
boBox1Change(Sender: TObject);
begin
 if ComboBox1.ItemIndex > MaxAvailable-
Level then
 begin
 ShowMessage('Этот уровень ещё не досту-
пен!');
 ComboBox1.ItemIndex := MaxAvailable-
Level;
 end
 else
 begin
 if CompletedLevels[ComboBox1.ItemIndex]
then
 begin
 ShowMessage('Этот уровень уже завер-
шён!');
 ComboBox1.ItemIndex := MaxAvailable-
Level;
 end
 else
 begin
 LoadSudoku(ComboBox1.ItemIndex);
 end;
 end;
end;
function TForm1.IsValid: Boolean;
var
 i, j, Num: Integer;
 RowSet, ColSet, BoxSet: set of 1..9;
begin
 // Проверяем, что все ячейки заполнены
 for i := 0 to 8 do
 begin
 for j := 0 to 8 do
 begin
 if not TryStrToInt(StringGrid1.Cells[j, i],
Num) then
 begin
 Result := False;
 Exit;
 end;
 end;
 end;
 // Проверяем строки и столбцы на уникаль-
ность чисел
 for i := 0 to 8 do
 begin
 RowSet := [];
 ColSet := [];
 for j := 0 to 8 do
 begin
 if not TryStrToInt(StringGrid1.Cells[j, i],
Num) then
 begin
 Result := False;
 Exit;
 end;
 if (Num < 1) or (Num > 9) then
 begin
 Result := False;
 Exit;
 end;
 if Num in RowSet then
 begin
 Result := False;
 Exit;
 end;
 Include(RowSet, Num);
 if Num in ColSet then
 begin
 Result := False;
 Exit;
 end;
 Include(ColSet, Num);
 end;
 end;
 // Проверяем квадраты 3x3 на уникальность
чисел
 for i := 0 to 2 do
 begin
```

```
 for j := 0 to 2 do
 begin
 BoxSet := [];
 for var k := 0 to 2 do
 begin
 for var l := 0 to 2 do
 begin
 if not TryStrToInt(StringGrid1.Cells[j * 3 + l, i
* 3 + k], Num) then
 begin
 Result := False;
 Exit;
 end;
 if Num in BoxSet then
 begin
 Result := False;
 Exit;
 end;
 Include(BoxSet, Num);
 end;
 end;
 end;
 end;
 Result := True;
end;
function TForm1.SolveSudoku(var Grid: array
of Integer): Boolean;
var
 Row, Col, Num: Integer;
begin
 if not FindUnassignedLocation(Grid, Row,
Col) then
 Exit(True);
 for Num := 1 to 9 do
 begin
 if IsSafe(Grid, Row, Col, Num) then
 begin
 Grid[Row * 9 + Col] := Num;
 if SolveSudoku(Grid) then
 Exit(True);
 Grid[Row * 9 + Col] := 0;
 end;
 end;
 Result := False;
end;
procedure TForm1.SpeedButton1Click(Sender:
TObject);
begin
LoadSudoku(ComboBox1.ItemIndex);
SpeedButton2.Enabled := True;
end;
procedure TForm1.SpeedButton2Click(Sender:
TObject);
begin
 if IsValid then
 begin
 ShowMessage('Решено верно!');
 Inc(LevelsCompleted);
 UpdateLevelsCompleted;
 SpeedButton3.Enabled := True;
 CompletedLevels[ComboBox1.ItemIndex] :=
True; //Помечаем текущий уровень как за-
вершённый
 // Устанавливаем максимальный доступный
уровень
 if ComboBox1.ItemIndex = MaxAvailable-
Level then
 Inc(MaxAvailableLevel);
 // Обновляем ComboBox1
 ComboBox1.Items.Clear;
 // Добавляем все уровни до максимального
доступного уровня
 if not CompletedLevels[0] then Com-
boBox1.Items.Add('Легкий')
 else ComboBox1.Items.Add('Легкий (Уро-
вень уже пройден)');
 if MaxAvailableLevel >= 1 then
 begin
 if not CompletedLevels[1] then Com-
boBox1.Items.Add('Средний')
 else ComboBox1.Items.Add('Средний (Уро-
вень уже пройден)');
 end
 else ComboBox1.Items.Add('Средний (Уро-
вень еще не доступен)');
 if MaxAvailableLevel >= 2 then
 begin
 if not CompletedLevels[2] then Com-
boBox1.Items.Add('Сложный')
 else ComboBox1.Items.Add('Сложный (Уро-
вень уже пройден)');
 end
 else ComboBox1.Items.Add('Сложный (Уро-
вень еще не доступен)');
 if MaxAvailableLevel >= 3 then
 begin
 if not CompletedLevels[3] then Com-
boBox1.Items.Add('Эксперт')
 else ComboBox1.Items.Add('Эксперт (Уро-
вень уже пройден)');
 end
```

```
 else ComboBox1.Items.Add('Эксперт (Уро-
вень еще не доступен)');
 if MaxAvailableLevel >= 4 then
 begin
 if not CompletedLevels[4] then Com-
boBox1.Items.Add('Экстримальный')
 else ComboBox1.Items.Add('Экстримальный
(Уровень уже пройден)');
 end
 else ComboBox1.Items.Add('Экстримальный
(Уровень еще не доступен)');
 // Устанавливаем выбранный элемент на
следующий доступный уровень
 ComboBox1.ItemIndex := Min(MaxAvailable-
Level, ComboBox1.Items.Count - 1);
 // Проверяем, если все уровни пройдены
 if ComboBox1.ItemIndex = 4 then
 begin
 ShowMessage('Все уровни пройдены!');
 end;
 end
 else
 ShowMessage('Решено неверно!');
 SpeedButton3.Enabled := True;
end;
procedure TForm1.UpdateLevelsCompleted;
begin
 Label1.Caption := 'Уровней пройдено: ' +
IntToStr(LevelsCompleted);
end;
procedure TForm1.SpeedButton3Click(Sender:
TObject);
var
 FlatSolution: array[0..80] of Integer;
 i, j: Integer;
begin
 for i := 0 to 8 do
 for j := 0 to 8 do
 FlatSolution[i * 9 + j] := Sudoku[i, j];
 if SolveSudoku(FlatSolution) then
 begin
 for i := 0 to 8 do
 for j := 0 to 8 do
 begin
 Solution[i, j] := FlatSolution[i * 9 + j];
 StringGrid1.Cells[j, i] := IntToStr(Solution[i,
j]);
 end;
 end
 else
 ShowMessage('No solution exists.');
end;
procedure TForm1.SpeedButton4Click(Sender:
TObject);
begin
Form3.Show;
Form1.Hide;
end;
function TForm1.FindUnassignedLocation(var
Grid: array of Integer; out Row, Col: Integer):
Boolean;
var
 Index: Integer;
begin
 for Index := Low(Grid) to High(Grid) do
 begin
 if Grid[Index] = 0 then
 begin
 Row := Index div 9;
 Col := Index mod 9;
 Exit(True);
 end;
 end;
 Result := False;
end;
function TForm1.IsSafe(var Grid: array of Inte-
ger; Row, Col, Num: Integer): Boolean;
var
 StartRow, StartCol, i, j: Integer;
begin
 for i := 0 to 8 do
 begin
 if (Grid[Row * 9 + i] = Num) or (Grid[i * 9 +
Col] = Num) then
 Exit(False);
 end;
 StartRow := (Row div 3) * 3;
 StartCol := (Col div 3) * 3;
 for i := 0 to 2 do
 for j := 0 to 2 do
 if Grid[(StartRow + i) * 9 + StartCol + j] =
Num then
 Exit(False);
 Result := True;
end;
end.
unit Unit2;
interface
uses
 Winapi.Windows, Winapi.Messages, Sys-
tem.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
```

```
 Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.Im-
aging.pngimage, Vcl.ExtCtrls,
 Vcl.ComCtrls, Vcl.Imaging.jpeg, Vcl.StdCtrls;
type
 TForm2 = class(TForm)
 Image1: TImage;
 ProgressBar1: TProgressBar;
 Timer1: TTimer;
 Label1: TLabel;
 procedure Timer1Timer(Sender: TObject);
 private
 { Private declarations }
 public
 { Public declarations }
 end;
var
 Form2: TForm2;
implementation
{$R *.dfm}
uses Unit3;
procedure TForm2.Timer1Timer(Sender:
TObject);
begin
 // Увеличиваем значение ProgressBar
 ProgressBar1.Position := ProgressBar1.Posi-
tion + 10;
 // Проверяем, достиг ли ProgressBar конца
 if ProgressBar1.Position >= 100 then
 begin
 Timer1.Enabled := False; // Останавливаем
таймер
 Form3.Show; // Показываем Form3
 Self.Close; // Закрываем текущую форму
 end;
end;
end.
unit Unit3;
interface
uses
 Winapi.Windows, Winapi.Messages, Sys-
tem.SysUtils, System.Variants, System.Classes,
Vcl.Graphics,
 Vcl.Controls, Vcl.Forms, Vcl.Dialogs,
Vcl.StdCtrls, Vcl.ExtCtrls,
 Vcl.Imaging.jpeg, Vcl.Buttons;
type
 TForm3 = class(TForm)
 Image1: TImage;
 SpeedButton1: TSpeedButton;
 SpeedButton2: TSpeedButton;
 SpeedButton3: TSpeedButton;
 Label1: TLabel;
 procedure FormShow(Sender: TObject);
 procedure SpeedButton1Click(Sender:
TObject);
 procedure SpeedButton3Click(Sender:
TObject);
 procedure SpeedButton2Click(Sender:
TObject);
 private
 { Private declarations }
 FForm2Shown: Boolean;
 public
 { Public declarations }
 end;
var
 Form3: TForm3;
implementation
{$R *.dfm}
uses
 Unit2, Unit1;
procedure TForm3.FormShow(Sender:
TObject);
begin
 if not FForm2Shown then
 begin
 Form2.ShowModal;
 FForm2Shown := True;
 end;
end;
procedure TForm3.SpeedButton1Click(Sender:
TObject);
begin
 Form3.Hide;
 Form1.Show;
end;
procedure TForm3.SpeedButton2Click(Sender:
TObject);
begin
ShowMessage('Разработал учащийся группы
ПЗТ-41 Линник Дмитрий' +#13+ 'Курсовой
проект:игровое приложение "Судоку"');
end;
procedure TForm3.SpeedButton3Click(Sender:
TObject);
begin
Close;
end;
end.
```