Einführung in Delphi

+

Übungsaufgaben zum Erlernen der Handhabung von Delphi

von Matthias Hühr unter Anleitung von Frau Prof. Ch.Wahmkow

Gliederung

1. **Einführung**

2. **Der Objektinspektor**
   - die wichtigsten Ereignisse
   - die wichtigsten Eigenschaften

3. **Die Objekthierarchie von Delphi**

4. **Der Aufbau einer Unit**

5. **Variablen**

6. **Schleifen**

7. **Arrays - Felder**

8. **Arbeiten mit Arrays - ein Sortieralgorithmus**

9. **Wichtige Prozeduren und Funktionen**

10. **Objekte der Symbolleisten**

11. **Tips zu Komponenten**

Literatur:

H. Erlenkötter
Delphi für Windows
RORORO Verlag 1996

E.Kaier
Delphi Essentials
Vieweg Verlag 1997

M.Ebner
Programmieren in Delphi
Addison- Wesley Verlag 1996

1. Einführung

Delphi ist die erste IDE (Integrated Development Environment), die aus mehreren Fenstern besteht. Das ist praktisch, da man die Einstellungen, die man im Objektinspektor vornimmt, gleich sehen kann.
Die IDE von Delphi besteht aus der Symbolleiste, dem Objektinspektor, dem Quelltexteditor und den Formularen als sichtbare Elemente sowie anderen wichtigen Werkzeugen.

- Auf den Formularen stellt man die Benutzeroberfläche für die eigene Anwendung zusammen
- Im Quelltexteditor schreibt man den Quelltext für das Programm in sogenannten Units
- Im Objektinspektor stellt man die Eigenschaften der Komponenten ein. Er teilt sich in drei Teile.

Ganz oben ist eine Combobox, in der man eine Komponente auswählen kann. Da sollte man immer einen Blick drauf werfen, um sicherzustellen, daß man sich in der gewünschten Komponente befindet. Darunter findet man zwei Registerkarten:
Auf dem linken Register stehen die Objekteigenschaften und auf dem rechten die Ereignisse.
Die Eigenschaften verändert man durch das Aktivieren der Felder.
Es gibt mehrere Möglichkeiten, um die Werte zu verändern. Entweder ist das ein Editierfeld, das man durch Anklicken aktiviert und dann Text oder Werte über die Tastatur eingibt. Das trifft zum Beispiel für Caption zu. Oder man aktiviert eine Combobox, aus der man aus verschiedenen vorgegebenen Werten einen auswählen kann. Das trifft für Fensterarten, Farben und Wahr-Falsch-Eigenschaften zu.
Es gibt desweiteren die Möglichkeit, daß ein rechteckiges Feld mit drei Punkten erscheint.
Wenn man dieses zweimal anklickt, erscheint ein Dialog. Dieser ist unter anderen für Schriftarten verfügbar.
Zum Register 'Ereignisse':
In ihr kann man die Prozeduren, die für die Objekte verfügbar sind, auswählen und editieren, indem man auf der rechten Seite das Feld hinter dem Ereignis zweimal anklickt und dann im Quelltexteditor den entsprechenden Quelltext hinzufügt.

- In der Symbolleiste findet man eine Auswahl der installierten Komponenten.

Eine Komponente platziert man auf das Formulur, indem man auf den Button in der Leiste drückt, der sich dann nach hinten bewegt und dann auf dem Formular platziert.
Sollte man sich verwählt haben, kann man, wenn man auf den Button mit dem Pfeil drückt, die Auswahl zurücknehmen.

Ein Programm startet man, indem man auf den Button mit dem grünen (violetten) Dreieck drückt.
Der Quelltext wird kompiliert und das Programm wird ausgeführt.
Der Compiler weist auf syntaktische Fehler hin, logische Fehler kann er leider nicht finden.

Sollte das Programm abstürzen oder eine Fehlermeldung erscheinen, die die Kompilierung betrifft (NICHT DEN QUELLTEXT !), wählt man im Objektinspektor den Befehl

START/PROGRAMM ZURÜCKSETZEN .

Der Quelltext wird bei der Kompilierung nicht abgespeichert, weshalb man den Quelltext öfter sichern sollte.
Man sollte die Hilfe benutzen, um Probleme zu lösen.
In ihr sind die verfügbaren Eigenschaften, Ereignisse und Funktionen der Objekte erklärt, sowie die wichtigsten Funktionen erläutert.
Es gibt auch Beispielprogramme, um Funktionen zu erläutern.
Die Objekte findet man, indem man ein T vor dem Namen setzt : TLabel .
Das T steht für Typ.

2. Der Objektinspektor

Im Objektinspektor stellt man die Eigenschaften des Objekts auf der Seite *Eigenschaften* ein und auf der Seite *Ereignisse* deklariert man die Prozeduren, die während der Laufzeit ausgeführt werden.

die wichtigsten Ereignisse sind:

OnChange:=Verändern der Werte

```
procedure TForm1.DirectoryListBox1Change(Sender: TObject);
begin
FileListBox1.Directory:=DirectoryListBox1.Directory;
end;
```

OnClick:= einmaliges Anklicken mit der Maus

```
procedure TForm1.Button1Click(Sender: TObject);
begin
OpenDialog1.Execute;
end;
```

OnClose:= Beenden des Programms

```
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
SaveDialog1.execute;
end;
```

OnDblClick := Doppelklick
OnHide:=Fenster unsichtbar machen, aber Programm nicht beenden
OnMouseDown:=Mautaste gedrückt

OnMouseMove:=Mauzeiger bewegt sich über Objekt

```
procedure TForm1.Button1MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
Application.HintColor:=clRed
end;
```

OnMouseUp:=Maustaste losgelassen

Onresize:=Fenstergröße verändern

```
procedure TForm1.FormResize(Sender: TObject);
begin
Panel1.Width:=Form1.Width div 2;
end;
```

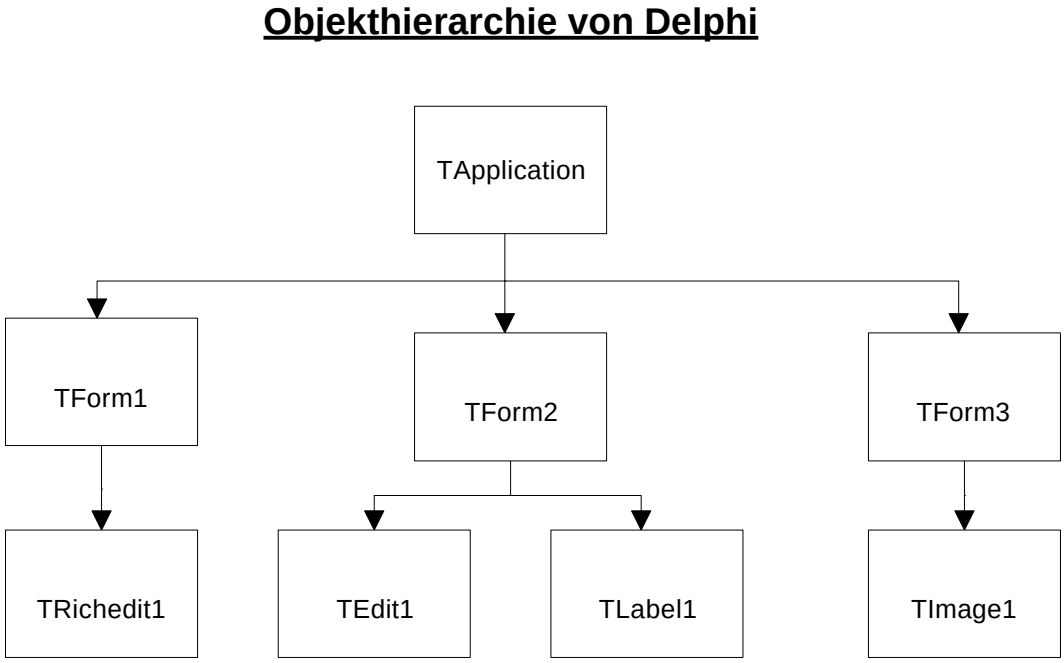
die wichtigsten Eigenschaften sind:

Align: Ausrichtung des Objects (z.B.oben,unten,links)
AutoScroll: ob Fenster selbst scrollt, wenn Inhalt größer als es selbst ist (True bzw. False)
Caption: Beschriftung
Height: Höhe des Objects
Enabled: Verfügbarkeit des Objektes (True bzw. False)
Font: Schriftart
Hint: Sprechblasen
Left: Position der linken oberen Ecke des Objekts im Fenster von links
ParentColor: die gleiche Farbe wie Eltern (True bzw. False)
ParentFont: die gleiche Schrift wie Eltern(True bzw. False)
ShowHint: Wert ob Hint gezeigt wird (True bzw. False)
Text: Inhalt von Textelementen; z.B. TEdit
Top: Position der linken oberen Ecke des Objekts im Fenster von oben
Width: Breite des Objekts
Visible: Wert ob sichtbar (True bzw. False)

Sämtliche Eigenschaften lassen sich auch programmtechnisch zuweisen; z.B:

```
if Zähler=1000 then Editfeld1.visible:=true;
```

3. Die Objekthierarchie von Delphi



Beispiel einer Hierarchie von Objekten in einem Programm mit 3 Fenstern.

Wenn ein Objekt auf eine Variable eines anderen Objektes zugreifen will, muß man bis zur Stelle zurückgehen, von der man beide Objekte erreichen kann.
Im oberen Beispiel für Edit1 und Label1 ist es Form2, für Richedit1 und Image1 ist es Application.
Um auf die Variablen zuzugreifen, muß man die Objekthierarchie zurückverfolgen.
Beispiele :

Label1.Caption:=Form3.Image1.Height;
RichEdit1.Lines.Strings[1]:=Form2.Edit1.Text;
Label1.Height:=Edit1.Height;
(Anmerkung: die Änderung erfolgt durch ein Objekt auf dem gleichen Formular.
Soll durch ein Objekt auf einem ganz anderen Formular diese Anweisungen erfolgen,
so muß auch das Formular im Pfad stehen)
Form2.Label1.Caption:=Form3.Image1.Height;

Wenn man in einer Hierarchie Werte verändert, die vererbbar sind, gelten sie für alle weiteren untergeordneten Objekte.
Beispiel:

Application.HintColor:=clRed;

Hier wird die Farbe der Sprechblasen rot.

4. Der Aufbau einer Unit

Units sind Sammlungen von Prozeduren, Funktionen, Variablen und/oder Konstanten. In Delphi gibt es jede Menge vordefenierter Units. Verwendet man eine Komponente aus einer solchen Unit, braucht man sich um deren Einbindung nicht zu kümmern. Delphi paßt auf, daß alles eingebunden wird.

Units kann man auch selbst erzeugen. Standardmäßig passiert das beim Erzeugen eines neuen Formulars. Eine gewisse Modularisierung ist also schon vorgegeben. Das ist günstig für die Wiederverwendbarkeit, Wartung und Pflege der Software und wirkt sich positiv auf die Kompilierungszeit aus.

Beim Erstellen einer Unit braucht man sich eigentlich nur auf das Wesentliche zu konzentrieren: **das Programm;** d.h. Name, andere Units, Interface usw. wird alles von Delphi erledigt.

unit Unit1;
Am Anfang steht UNIT und der Name der Unit. Der Name der Unit wird spätestens beim Abspeichern festgelegt. Delphi bindet die Unit dann gleich ins Hauptprogramm ein. Bitte **nichts „per Hand" manipulieren**, Delphi nimmt das übel und es passiert leicht, daß gar nichts mehr zusammen paßt.

interface
Die im Interfaceteil einer Unit stehenden Deklarationen können von anderen Units benutzt werden.
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
FileCtrl, StdCtrls, ExtCtrls;
Im USES-Teil einer Unit stehen die Units, die geladen werden, um Funktionen und Prozeduren, die in diesen Units stehen, benutzen zu können.
Delphi fügt die meisten Units selbständig hinzu, wenn Komponenten plaziert werden.
Zum Beispiel stehen in *Windows* die Funktionen, die für die Fenstertechnik zuständig sind.

```
type
  TForm1 = class(TForm)
    Image1: TImage;
    Panel1: TPanel;
    FileListBox1: TFileListBox;
    DriveComboBox1: TDriveComboBox;
    DirectoryListBox1: TDirectoryListBox;
    procedure FileListBox1Change(Sender: TObject);
```

Unter TYPE deklariert Delphi die Komponenten, die im Programm verwendet werden.
Sie werden durch Vererbung erstellt. *Panel1* stammt vom Typ *TPanel* ab und
es besitzt alle Eigenschaften, die durch den Typ *TPanel* vorgegeben sind.
Es werden hier auch die Prozeduren, die unter *Implementation* programmiert werden, deklariert.
In diesem Fall :**procedure FileListBox1Change.**
Um die TYPE-Anweisung braucht man sich in der Regel auch nicht zu kümmern, bloß muß man aufpassen, daß zu jeder deklarierten Prozedur oder Funktion auch ein Programm gehört.
Also nicht einfach den Quelltext einer Prozedur löschen, sondern auch deren Deklaration !

```
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;

var
  Form1: TForm1;
```

Unter VAR stehen die globalen Variablen. Das sind die Variablen, die für die gesamte Unit gültig sind.

implementation
{$R *.DFM}

```
procedure TForm1.FileListBox1Change(Sender: TObject);
begin
if filelistbox1.filename<>''then
image1.picture.loadfromfile(filelistbox1.filename);
end;
```

Prozeduren werden durch das Schlüsselwort *procedure* eingeleitet. Sie können einen eigenen VAR-Teil besitzen, d.h. die hier deklarierten Variablen gelten nur in der Prozedur, also lokal.
Der Anweisungsteil steht zwischen **Begin** und **end;**

end.
Das END mit einem Punkt stellt das Ende der Unit dar.

5. Variablen

Variablen sind Bezeichner für Werte. Die Variablen werden im Variablendeklarationsteil der Unit, der Prozedur oder der Funktion deklariert. Dieser wird mit **VAR** gekennzeichnet.
Bsp:

```
VAR
X, Y, Z: Real;
  I, J, K: Integer;
Matrix: array[1..10, 1..10] of Real;
```

Die Variablen können global deklariert werden, sie gelten dann in der gesamten Unit.
Dieses macht man an der Stelle, wo auch schon Form1deklariert ist (oberhalb von Implementation).

```
...
var
  Form1: TForm1;
 X, Y, Z: Real;
  I, J, K: Integer;
Matrix: array[1..10, 1..10] of Real;

implementation
...
```

Man kann Variablen auch lokal deklarieren. Sie gelten dann nur in einer Prozedur.

```
...
procedure TForm1.Timer1Timer(Sender: TObject);
var
i:integer;
begin
  ...
end;
...
```

Man kann eine Variable mit dem gleichen Namen global und lokal deklarieren.
Die lokalen Variablen werden in der entsprechenden Prozedur bevorzugt, wenn es eine globele Variable mit gleichem Namen gibt.

Die Wertzuweisungen der Variablen erfolgen im Programm

Dort schreibt man den Variablennamen und **:=** und den neuen Wert.
Beispiel:

```
....
Var
x:integer;
Dateiname:string;
Begin
x:=10;
Dateiname:='C:\test.txt';
end;
```

...

Zu beachten ist, daß Strings in Anführungszeichen stehen müssen.

Die wichtigsten Variablentypen:
Real -Gleitkommazahlen, 8 Stellen Genauigkeit
Double - doppelt genaue Gleitkommazahlen, 16 Stellen Genauigkeit
String -Zeichenketten
Integer -Ganze Zahlen von -32367 bis +32368
Boolean -wahrer Ausdruck
Char -Zeichen
Longint -Ganze Zahlen
und neu in Delphi:
Variant - darauf kann alles stehen, Delphi erkennt selbständig den Datentyp, der auch während der Laufzeit geändert werden kann.
Aber aufpassen, Variant hat den Nachteil, daß er 265 Byte groß ist, also viel Speicher braucht und in der Verarbeitung sehr langsam wird.
Currency- spezieller Datentyp für die Finanzrechnung und bietet eine erhöhte Genauigkeit

Man kann die Datentypen nicht beliebig zuweisen, da sie alle unterschiedlichen Speicherplatz benötigen.
z.B.:
eine normale Realzahl benötigt 6 Byte Speicherplatz. Eine Integerzahl nur 2 Byte. Die Realzahl ist für den Integerspeicherplatz zu **fett.**

```
Var
  i: integer;
  r: real;
begin
 r:=3.14;
 i:=r;     ist falsch !!!
 i:=5/4;  ist falsch, da das Ergenbis reel ist !
```

6. Schleifen

In fast jedem Programm, welches Werte berechnet, kommen Schleifen vor.
Diese sind da, um zum Beispiel eine bestimmte Anzahl von Berechnungen durchzuführen,
oder solange zu rechnen, bis ein Wert so groß ist, wie er gefordert ist.

FOR - Zählschleife
Bei einer For-Schleife werden die Anweisungen sooft wiederholt, wie es im Schleifenkopf deklariert ist. In Pascal können die Schleifen nur in Einerschritten gezählt werden.
In dem Beispiel zählt die Schleife von 1 bis 10 und das Ergebnis ist am Ende 10.

```
...
x:=0;
For i:=1 to 10 Do
  Begin
    x:=x+1;
 End;
...
```

Es geht auch abwärts:

```
...
x:=0;
For i:=10 downto 1 Do
Begin
x:=x+1:
End;
...
```

REPEAT- Schleife
In einer Repeat-Schleife werden die Anweisungen solange wiederholt, bis der am Ende stehende Ausdruck wahr ist.

```
...
x:=0
repeat
   x:=x+1;
Until x=10;
...
```

Die Schleife wird 10 mal durchlaufen.

WHILE- Schleife
In einer While-Schleife werden die Anweisungen sooft wiederholt, solange der im Schleifenkopf stehende Ausdruck wahr ist.

```
...
x:=0;
While x <= 10 do
  x:=x+1;
...
```

Die Schleife wird 11 mal durchlaufen.

Bei allen Schleifen ist man nicht gezwungen, diese immer bis zum bitteren Ende zu durchlaufen.
Mit den Befehlen *break* und *continue* kann man sie auch direkt verlassen bzw. an ihren Anfang springen.
Break bricht die Schleife ab und setzt die Bearbeitung hinter der Schleife fort.
Continue springt zum Anfang der Schleife. Hier wird der nächste Schleifendurchlauf gestartet

7. Arrays - Felder

Mit Arrays kann man viele mathematische Sachverhalte beschreiben.
Ein eindimensionales Array ist zum Beispiel ein einfacher Zahlenstrahl.
Einen zweidimensionalen Array kann man sich als Schachbrett vorstellen,
bei dem die zugewiesenen Werte die Figuren auf dem Brett sind.
In diesem Fall wird das Array in der Variablenvereinbarung als Array deklariert, dessen Elemente Zeichenketten sind:

```
Var
Schachbrett:array[1..8,1..8]of string;
   im Programm wird dann z.B. der Wert zugewiesen:
Schachbrett[1,4]:='Dame';
```

Arrays werden in der Variablendeklaration im Programm deklariert
Beispiele:

```
Var
eindimensional:array[1..100]of integer;{maximal 100 Werte möglich}
zweidimensional:array[1..20,1..20]of double;{maximal 400 Werte (20*20)}
dreidimensional:array[1..10,1..10,1..10]of string;{maximal 1000 Werte(10*10*10)}
```

Dabei ist im ersten Beispiel die 1 der erste Index und die 100 der letzte Index.
Beachte:
Für Arrays nimmt man eckige Klammern!
Beide Zahlen werden durch **zwei** Punkte getrennt !
Bei mehrdimensionalen Arrays werden die Dimensionen durch ein Komma getrennt!

Die **Wertezuweisung** erfolgt im Programm über das Ansprechen der **einzelnen Feldelemente**:
Feldname[Index].

```
Zensur[1]:=2;
Zensur[2]:=1;          (wenn Zensur:array[1..n]of integer ist)
```

Zensur[3]:=3;

name[1]:=’alf’ (wenn name:array[1..n]of string ist , Strings immer in Anführungszeichen setzen!)

```
for i:=1 to 10 do
   a[i]:= i;
```

8. Arbeiten mit Arrays - ein Sortieralgorithmus

Variablendeklaration:

```
Var
Wert:array[1..1000]of double;
i,j:integer;
Hilfe:double;
```

Das Array so dimensionieren, wie es gebraucht wird oder wenn maximale Anzahl der Werte nicht bekannt ist, einen großen Wert zur Sicherheit wählen (Überdimensionieren)
WERT NICHT ÜBERTREIBEN - ARBEITSSPEICHER WIRD BENÖTIGT !
...
Wertzuweisung irgendwo im Programm:

```
Wert[1]:=9;
Wert[2]:=5;
```

etc.
...

An einer anderen Stelle im Programm (die Werte wurden schon belegt) erfolgt die Sortierung des Arrays.
Beispielalgorithmus:
Der Algorithmus tauscht die nebeneinanderstehenden Elemente im Array aus, indem er die Werte vergleicht, Ist der Vorgänger kleiner als sein Nachfolger, wird die Position ausgetauscht. Auf diese Art und Weise wird ein großer Wert bei jedem Schleifendurchlauf nach vorn geschoben, ein kleiner nach hinten.

```
For j:=1 to n do
        For i:=1 to (n-1) do
                if Wert[i]<Wert[i+1] then
                        begin
                                Hilfe:=Wert[i];
                                Wert[i]:=Wert[i+1];
                                Wert[i+1]:=Hilfe;
                        end;
        end;
end;
```

Wertzuweisung beim Durchlauf der Schleife:

i	Wert[1]	Wert[2]	Wert[3]	Wert[4]	Wert[5]	Wert[6]	Wert[7]	Wert[8]	Wert[9]	Wert[10]
0	9	5	7	6	4	3	8	1	10	2
1	5	7	6	4	3	8	1	9	2	10
2	5	6	4	3	7	1	8	2	9	10
3	5	4	3	6	1	7	2	8	9	10
4	4	3	5	1	6	2	7	8	9	10
5	3	4	1	5	2	6	7	8	9	10
6	3	1	4	2	5	6	7	8	9	10
7	1	3	2	4	5	6	7	8	9	10
8	1	2	3	4	5	6	7	8	9	10
9	1	2	3	4	5	6	7	8	9	10
10	1	2	3	4	5	6	7	8	9	10

Anm.: Daß die Ordnung schon nach dem siebenten Durchlauf fertig war, liegt an den Werten.

9. Wichtige Prozeduren und Funktionen

VAL - Umwandlung von Strings in Integer- bzw. Doublewerte
Bsp.:
Aus dem Text von Edit1 wird eine Integerzahl ermittelt.

```
var
  I, Code: Integer;
begin
  Val(Edit1.Text, I, Code);
 end;
```

Wenn der Wert der Variablen Code = 0 ist, ist kein Fehler bei der Umwandlung aufgetreten.
Wenn er <> 0 ist, ist eine fehlerhafte Eingabe erfolgt und der Wert von I wird als 0 angenommen.
Der Wert von Code ist die Stelle im String, der fehlerhaft ist.

INTTOSTR - Umwandeln einer Integerzahl in einen String
Bsp.:
Eine Zahl wird durch ein Label ausgegeben.

```
var
x:integer;
begin
  x:=5;
  Label1.Caption:='Die Zahl heißt : ' + inttostr(x);
end;
```

Strtoint - Umwandeln eines Strings in eine Integerzahl
Bsp.:

```
var
a:integer
begin
  a:=strtoint(TrackBar1.Position);
end;
```

Floattostr - Umwandeln einer gebrochenen Zahl in einen String
Bsp:

```
var
x:double;
begin
  Edit1.Text:=floattostr(x);
end;
```

Strtofloat - Umwandeln eines Strings in eine gebrochene Zahl
Bsp:

```
var
f:double;
begin
  f:=strtofloat(edit1.text);
end;
```

Copy - Kopieren von einer Zeichenkette aus einem String
Bsp.:
Aus dem String 'ABCDEF' wird der Variablen S der Wert 'BCD' zugeordnet.
Die erste Zahl ist die Position des ersten Zeichen und die zweite Zahl die Anzahl der folgenden Zeichen.

```
var S: string;
begin
  S := 'ABCDEF';
  S := Copy(S, 2, 3);
end;
```

Timetostr - Ermitteln der Uhrzeit
Bsp.:
```
Label1.Caption:=Timetostr(time);
```

Datetostr - Ermitteln des Datums
Bsp.:
```
Label1.Caption:=Datetostr(date);
```

Div - Division von Integerzahlen (Stellen vor dem Komma)
Bsp.:
```
var
x,y,z:integer;
begin
  x:=5;  y:=4;
  z:=x div y;    {Das Ergebnis ist z=1}
end;
```

Mod - Division von Integerzahlen (Stellen nach dem Komma)
Bsp.:
```
var
a,b,c:integer;
begin
  c:=a mod b;
end;
```

Round - Runden von Double-Variablen bzw. Umwandeln von Double- in Integerwerte
Bsp.:
```
var
a:integer;
b:double;
begin
  a:=round(b);
end;
```

Diskfree - Ermitteln des freien Speicherplatzes auf einem Laufwerk
Bsp.:
```
var
  S: string;
begin
  S := IntToStr(DiskFree(0) div 1024) + ' KByte frei.';
 Label1.Caption:=S;
end;
```
0 ist das aktuelle Laufwerk, 1 ist A ,2 ist B ,3 ist C etc.

Disksize - Ermitteln der Laufwerksgröße
Bsp.:
```
var
  S: string;
begin
  S := IntToStr(DiskSize(0) div 1024) + ' KByte Kapazität.';
  Label1.Caption:=S;
end;
```

Extractfilename - Abschneiden des Pfades aus dem Dateinamen
Bsp.:

```
Form1.Caption := 'Datei :  '+ ExtractFileName(OpenDialog1.FileName);
```

Extractfilepath - Ermitteln des Verzeichnisses, in dem sich eine Datei befindet
Bsp.:
```
Label1.Caption:=’Verzeichnis : ‘+ExtractFilePath(Opendialog1.FileName);
```

Extractfilesextension - Ermitteln des Dateityps
Bsp.:
```
Dateiextension := ExtractFileExt(Opendialog1.FileName);
```

Messagedlg - Dialog mit Wahlmöglichkeiten
Bsp.:
```
if MessageDlg('Programm beenden?',
   mtInformation, [mbYes, mbNo], 0) = mrYes then
  begin
    MessageDlg('Programm wird beendet.', mtInformation,
      [mbOk], 0);
    Close;
  end;
```

Showmessage - einfaches Nachrichtenfenster
Bsp.:
```
Begin
{$i-}
Chdir(‘A:\’);
{$i+}
if ioresult =21 then
showmessage(‘Keine Diskette im Laufwerk A’);
end;
```

Chdir - Wechsel in ein Laufwerk bzw. Verzeichnis
Beispiel siehe oben

Getdir - Ermitteln des Laufwerkes
Bsp.:
```
var
   s : string;
 begin
   GetDir(0,s);
   MessageDlg('Aktuelles Laufwerk und Verzeichnis: ' + s, mtInformation, [mbOk] , 0);
 end;
```
0 ist aktuelles Laufwerk 1 ist A 2 ist B etc.

Setfocus - Objekt erhält Eingabefocus
Bsp.:
```
Edit1.setfocus;
```

Loadfromfile - Datei laden
Bsp.:
```
Memo1.Lines.LoadFromFile(OpenDialog1.FileName);
```

Savetofile - Datei speichern
Bsp.:
```
RichEdit1.Lines.SaveToFile(Dateiteiname);
```

10. Objekte der Symbolleisten

Objekte der Standard - Symbolleiste

MainMenu — Mainmenu ist die Menüzeile,die bei den meisten Programmen verfügbar ist. Es ist ein Objekt, das bei der Entwicklung Einträge und Prozeduren erhält, indem man ein Mainmenu auf dem Formular platziert und dann durch den Menueditor, der durch Doppelklick aktiviert wird, gestaltet. (Beispiel ist in Übung 2 vorhanden) Man kann zum Anlegen eines Menüs auch einen der vielfältigen Assistenten benutzen.

PopupMenu — Das Popupmenu ist das Menü, welches erscheint, wenn man die rechte Maustaste drückt. Man kann jedem sichtbaren Objekt ein Popupmenu zuordnen. Das macht man, indem man im Objektinspektor bei dem jeweiligen Objekt unter dem Eintrag Popupmenu, das entsprechende auswählt.
Die Gestaltung ist ähnlich dem Mainmenu.

Label — Label sind Beschriftungsfelder, welche man zur Erläuterung von Objekten und zur Gestaltung der Benutzeroberfläche nutzen kann.

Edit — Editfelder sind Eingabefelder, wo man Werte über die Tastatur eingeben kann. Es sind **immer** Strings, die eingegeben und ausgegeben werden können. Um die Zahlenwerte verwenden zu können, muß man sie vorher umwandeln.
Beispiel für natürliche Zahlen :

```
a:=inttostr(edit1.text);
```

Beispiel für gebrochene Zahlen :

```
b:=inttofloat(edit2.text);
```

Diese Beispiele setzen voraus, daß die Werte auch dementsprechend eingegeben werden.
Besser wäre die Funktion VAL.
Um die Ergebnisse wieder auszugeben nimmt man die Funktionen strtoint bzw. floattoint.

Memo — Memos sind Textfelder, die ganze Seiten anzeigen können. Man kann es für einfache Texteditoren einsetzen.
(Beispiel in Aufgabe 3)

Button — Das sind die Befehlsschaltflächen, denen man Aufgaben (Ereignisse) zuordnen kann, wie zum Beispiel das Programm zu beenden.

CheckBox — Auswahlfeld, bei dem man zum Beispiel einstellen kann, ob verschiedene Funktionen im Programm verfügbar sind.

RadioButton — Auswahlfeld, welches bei einer Gruppe, die sich in einer Radiogroupbox befindet, nur bei einem Radiobutton die Aktivierung zulässt.
Ein Anwendungsbeispiel wäre, wie in vielen Programmen, die Identifizierug des Nutzers bei Bestellformularen

die Auwahl bei Geschlecht : männlich oder weiblich .

ListBox — Mit der Listbox kann man Auswahllisten erstellen.
Beispiele sind die Fontdialoge bei Textverarbeitungsprogrammen.

ComboBox — Das sind auch in der Praxis oft verwendete Objekte.
Beispiele sind die Schriftart bzw Schriftgrößeeinstellungen in Textverarbeitungsprogrammen.

ScrollBar — Scrollbalken, den man für scrollbare Objekte innerhalb des Fensters nehmen kann.

GroupBox — In einer Groupbox kann man viele Objekte, die zusammengehören zusammenfassen, um die Übersichtlichkeit zu erhöhen.

RadioBox — In einer Radiobox platziert man zusammengehörende Radiobuttons.
In einer Radiobox kann nur ein Radiobutton aktiv sein.

Panel — Panels sin Kontainer für Kontrollelemente oder andere Objekte.
Mit Panels kann man auch die Seiteneinteilung gestalten.

Objekte der Windows 95 - Symbolleiste

TabControl — Element für mehrseitige Dialoge.

PageControl — Ein Pagecontrol nimmt man für mehrseitige Dialoge, wie sie in vielen großen Programmen vorkommen.

TreeView — Das ist ein Element, das man für Baumdiagramme nehmen kann.

ListView — Listview nimmt man für veränderliche Darstellungen von Dateien, ob als Liste oder Icon.

ImageList — Objekt, in dem man kleine Bilder speichern kann.
Unter anderen Icons für die Listviewkomponente.

HeaderControl — Für verstellbare Breiten wie bei Eigenschaftsbeschriftungen bei Listen.

RichEdit — Objekt wie das Memofeld. Es hat aber mehr Funktionalität.

Drag und Drop mit dem Text ist möglich.

StatusBar — Die Statusbar befindet sich bei vielen Programm unten. In ihr kann man bestimmte Werte ausgeben, die der Anwender zu Information erhält.

TrackBar — Trackbar ist ein Schieberegler, mit dem man Werte einstellen kann.

ProgressBar — Die Progressbar ist ein Fortschrittsanzeiger, der in vielen Programmen vorkommt. Mit ihr kann man anzeigen, inwieweit eine Datei gelesen wurde etc.

UpDown — Updown ist ein Schalter, mit dem man Integerwerte einstellen kann.

HotKey — Das ist ein Tastaturkürzelsteuerelement.

Objekte der Zusätzlich - Symbolleiste

BitBtn — BitBtn ist ein Button, dem man auch ein Bitmap zuweisen kann.

SpeedButton — Speedbuttons sind Buttons, die man mit der Eigenschaft GroupIndex gruppieren kann, so daß nur ein Button der Gruppe als gedrückt angezeigt werden kann.

MaskEdit — Beim Maskedit kann man eine Eingabemaske definieren, so daß zum Beispiel nur Datumsangaben eingegeben werden können.

StringGrid — Das ist ein Element, in dem man Strings eingeben und bearbeiten kann. Es wird zum Beispiel in Tabellenkalkulationen verwendet.

DrawGrid — Das ist der Vorgänger vom StringGrid. Es hat aber weniger Eigenschaften.

Image — Mit dem Imageobjekt kann man Bitmaps und Icons anzeigen lassen. Andere sind von Delphi aus nicht möglich, außer, wenn man den entsprechenden Code für die anderen Dateitypen besitzt.

Shape Das sind graphische Objekte, mit denen man Kreise,Rechtecke und weitere Formen zeichnen kann.

Bewel Ein Bevel ist ein Gestaltungselement für Rahmen etc.

ScrollBox In einer Scrollbox kann man Objekte platzieren , die größer als das Formular sind.

Objekte der Dialog - Symbolleiste

OpenDialog Das ist ein Dialog zum Öffnen von Dateien.

SaveDialog Das ist der Dialog zum Speichern von Dateien

FontDialog Das ist ein Dialog zum Auswählen von Schriftatributen.

ColorDialog Ein Dialog für die Auswahl von Farben.

PrintDialog Das ist der Dialog der für das Ausdrucken von Dateien zuständig ist.

PrinterSetupDialog Dialog zum Einrichten des Druckers.

FindDialog Das ist ein in Textverarbeitungen üblicher Dialog zum Finden von Wörtern.

ReplaceDialog Dialog zum Suchen und Austauschen von Wörtern.

Objekte der System - Symbolleiste

Timer Der Timer ist eine Komponente für sich zeitlich widerholende Prozeduren.

 PaintBox In einer PaintBox kann man graphische Objekte erzeugen.

 FileListBox Eine Filelistbox zeigt Dateien eines Verzeichnisses an.

 DirectoryListBox Die Directorylistbox zeigt Verzeichnisse an.

 DriveComboListBox Box zum Anzeigen von Laufwerken.

 FilterComboBox Das ist eine Box zur Auswahl von Dateitypen die angezeigt werden können.

 MediaPlayer Das ist eine Komponente zum Abspielen von Bild- und Tondateien.

OleControl Komponente zum Einbetten von Objekten.

Anmerkung:
Es gibt noch weitere Komponenten, auf die ich nicht weiter eingehe, da sie nicht notwendig für die Grundkenntnisse sind. Es sind auch in jeder Delphiversion andere Komponenten verfügbar . Außerdem kann man selbst eigene Komponenten hinzufügen.

11. Tips zu Komponenten

TStatusBar

Um eine Statusbar zu gestalten, wählt man im Objektinspektor den Eintrag *Panels* an.
Ein Dialog erscheint dann. Dort wählt man den Eintrag **Neu**, um ein Feld hinzuzufügen.
Die Einstellungen für das Verhalten, wie Schriftausrichtung können im Dialog verändert werden.
Um die Uhrzeit im linken Panel darzustellen, fügt man einen Timer dem Programm hinzu.
Dann schreibt man folgende Prozedur :

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
StatusBar1.Panels[0].Text:=timetostr(time);
end;
```

Es wird ab 0 gezählt!

TListBox

Wenn man einen Eintrag von einer Listbox zu einer anderen mittels eines Buttons verschieben will, kann man folgendes Beispiel verwenden:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
if listbox1.itemindex>= 0 then
begin
```

```
listbox2.items.add(listbox1.items.strings[listbox1.itemindex]);
listbox1.items.delete(listbox1.itemindex);
listbox1.setfocus;
end;
end;
```

TListView

Um in einer Listview die Dateien eines Verzeichnisses anzuzeigen und nicht viel Quelltext eingeben will, kann man folgendes machen:
Man plaziert hinter dem Listview eine Filelistbox und verknüpt eine Directorylistbox mit der Filelistbox und schreibt diese Prozedure.

```
procedure TForm1.FileListBox1Change(Sender: TObject);
var NewItem : TListItem;
 i, x:integer;
begin
x:=(filelistbox1.items.count);
listview1.items.clear;
for i:=0 to x-1 do
if i>=0 then
begin
  NewItem := ListView1.Items.Add;
  NewItem.Caption := filelistbox1.items.strings[i];
end;
end;
```

Um auf die Einträge zuzugreifen, nimmt man folgendes Beispiel:

```
procedure TForm1.ListView1DblClick(Sender: TObject);
begin
 if listview1.selected = nil then exit;
form2.jpegimage1.filename:=listview1.selected.caption;
end;
```

Anmerkungen: TJpegImage ist keine Originalkomponente
Icons muß man über eine Imagelist einfügen
das Beispiel ist nur geeignet,wenn man nur einen Dateityp darstellen will