Arbeitsmaterial

Kleine Einführung in die Programmierumgebung Delphi

Autor: Barbara Flütsch
http://sos-mathe.ch

Speichern

Speichern Sie jedes Projekt in einem neuen Ordner, dem Sie den Namen der Übung geben!
Wählen Sie dabei immer die Option "Alles Speichern"!

Vergessen Sie die Anweisungen im Buch!

Öffnen eines Projektes

Doppelklicken Sie das Delphisymbol mit dem Schriftzug "Project1" (in Delphi 3)

Ausführen und Beenden eines Projekts

Doppelklicken Sie das Fackelsymbol mit dem Schriftzug "Project1"
oder, wenn das Projekt geöffnet ist: Taste F9

Beenden mit Klick ins Schliessfenster oben rechts oder mit Alt+F4. **Nicht vergessen!**
w

Tipps

Klick ins Formular:	Caption:	Titel ändern
	Color:	Farbe wählen
	Font:	Schriftart, -grösse, -farbe etc für alle Elemente des Formulars wählen

Markierte Taste (Button) oder Label (Textelement):	Caption:	Titel
	Font:	Schriftart etc

Wenn kein Objekt ausgewählt ist, lassen sich Schriftart etc. für alle folgenden Objekte einstellen.

Mehrere Objekte ausrichten: alle markieren (mit Klick und Shift+Klick oder mit gedrückter Maustaste einen Rahmen darum ziehen)

Drücken der rechten Maustaste auf eines der Objekte liefert ein Kontextmenü mit dem Titel "Ausrichten"

Mehrere gleiche oder fast gleiche Objekte im Formular und Texte im Editor:

Kopieren: Ctrl + c und einfügen: Ctrl + v

Noch rascher: Text im Editor markieren und bei gedrückter Ctrl-Taste verschieben!

VARIABELN

Variabeln können wie in der Mathematik mit einem einzigen Buchstaben, aber auch mit einem ganzen Wort bezeichnet werden (Zahl, Kapital, Zins); Sie dürfen aber keine von Delphi reservierten Wörter verwenden (Label, Text)

Variabeln müssen definiert werden:

Diese Definition kann man vor der 1. Prozedur anbringen, dann gelten die Variablen für alle folgenden Prozeduren. Man nennt sie globale Variable.

```
Var
   Form1: Tform1;
   a,b,c: Integer;
   Name: String;
```

Mann kann sie aber auch innerhalb einer Prozedur definieren (lokale Variablen).

```
procedure Tform1.Button1.Click(Sender:Tobject);
Var
   i, j: Integer;
begin
   . . .
```

Standardmässig sind Zahlvariable mit dem Wert 0 versehen, Zeichenvariable sind leer.
Sie können aber eine Variable schon bei der Deklaration mit einem Anfangswert versehen:

```
Var
   i: Integer=5;
   s: String='Hallo';
```

Genau so (und an denselben Stellen) könnten Sie Konstanten definieren, die später ihren Wert wieder ändern dürfen:

```
const
   i: Integer=5;
   s: String='Hallo';
```

Konstanten, die während der ganzen Anwendung unveränderlich sind definiert man als:

```
const
   pi=3.14;
   Titel='Harry Potter';
   Preis=25*4.5;
```

Eine Auswahl der verschiedenen Variablentypen:

Es gibt in Delphi verschiedene Arten von Variabeln, die verschieden viel Speicherplatz belegen.

Ganze Zahlen:

Integer	Ganze Zahlen von –2'147'483'648 bis 2'147'483'648 ($=2^{31}$)
ShortInt	Ganze Zahlen von -128 bis 128 ($=2^{7}$)
Byte	Ganze Zahlen von 0 bis 255

Reelle Zahlen:

Real	Reelle Zahlen von $2.9 \cdot 10^{-39}$ bis $1.7 \cdot 10^{38}$ mit einer Genauigkeit von 11-12 Stellen
Single	Reelle Zahlen von $1.5 \cdot 10^{-45}$ bis $3.4 \cdot 10^{38}$ mit einer Genauigkeit von 7-8 Stellen
Double	Reelle Zahlen von $5 \cdot 10^{-324}$ bis $1.7 \cdot 10^{308}$ mit einer Genauigkeit von 15-16 Stellen
Extended	mit einer Genauigkeit von 19 bis 20 Stellen

Benützen Sie lieber die Typen Single und Double, Real braucht wesentlich mehr Rechenzeit.

Zeichenvariablen:

Char	1 Buchstabe
String	eine Zeichenkette (bis 2 Gigabyte Zeichen)

Boole'sche Typen:

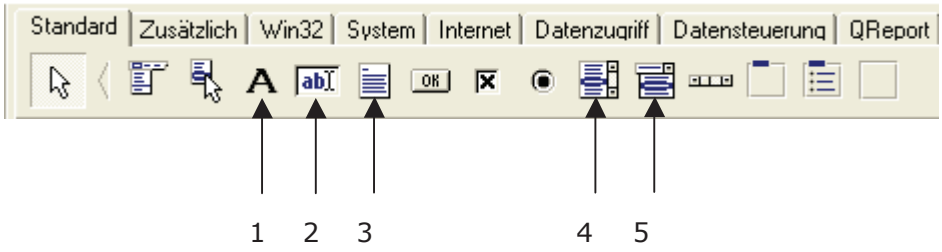
Boolean	hat nur die Werte True und False

Delphi liest und schreibt auf Labels und in Editfeldern nur Zeichenketten, die man vor einer Rechnung umwandeln muss:

StrToInt	Zeichenkette → ganze Zahl
IntToStr	Ganze Zahl → Zeichenkette
StrToFloat	Zeichenkette → reelle Zahl
FloatToStr	reelle Zahl → Zeichenkette

FloatToStrF(Zahl,ffGeneral,5,0) schreibt die Zahl auf 5 Stellen genau.

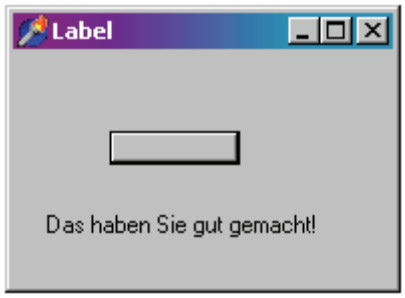
Text und Eingabefelder



Label (1)

Ein Label nimmt irgendwelche einzeiligen Texte auf; der sichtbare Text wurde im Objektinspektor unter Caption eingegeben.

Drücken der Taste lässt den sichtbaren Text verschwinden und – im Label2 – den neuen Text erscheinen.



Befehle

```
Label2.Caption:='Das haben Sie gut gemacht';
Label1.Caption:=' ';
```

(Denselben Effekt können Sie natürlich auch mit der Eigenschaft "visible" erreichen.)

Edit-Feld (2)

Um während des Programmablaufes Text **einzugeben**, benötigen Sie ein Edit-Feld;

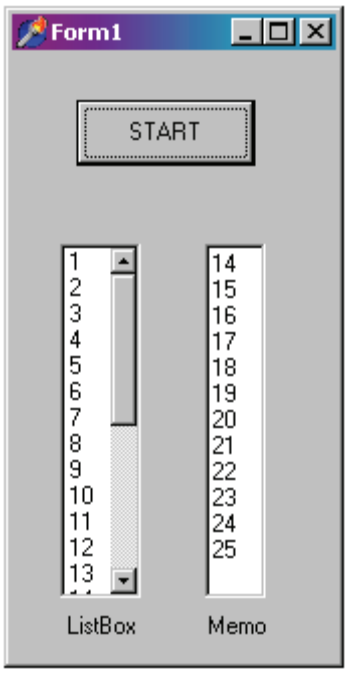
Sie weisen den eingegebenen Text mit dem folgenden Befehl der Stringvariablen "Satz" zu:

```
Satz:=Edit1.Text;
```

Falls die Variable x vom Typ Integer ist heisst der Befehl:

```
x:=StrToInt(Edit1.Text);
```

Memo (3), ListBox (4) und ComboBox (5)



Unterschied zwischen ListBox und Memo

Beide Male werden die Zahlen von 1 bis 25 aufgelistet; in der ListBox kann man blättern und alle Zahlen betrachten, im Memo sieht man nur noch die letzten davon.

Befehle zur Eingabe:

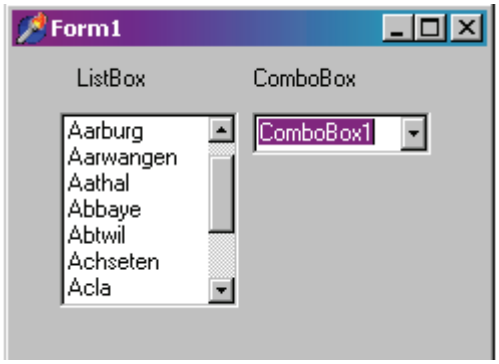
ListBox.Items.Append(IntToStr(n));

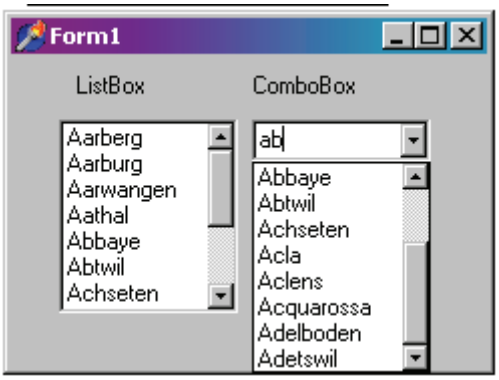
Memo1.Lines.Add(IntToStr(n));

Befehle zum Löschen:

ListBox.Items.Clear;

Memo1.Lines.Clear;





Anwendung von ListBox und ComboBox

Beide können mit einer Liste von Einträgen gefüllt werden: Wählen Sie im Objektinspektor die Eigenschaft "Items", klicken Sie auf  und geben Sie die Liste ein.

Die Listeneinträge werden intern nummeriert (beginnend bei 0);

mit z. B.

if Listbox1.ItemIndex=3 **then . . .**

bzw.

if ComboBox1. ItemIndex=3 **then . . .**

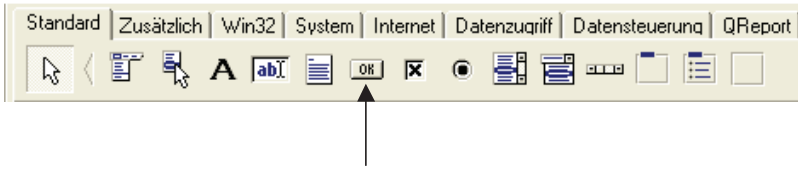
können Sie bei Wahl des 4. Eintrags eine Handlung auslösen.

Klicken Sie bei der ComboBox auf das Dreieck, um die Liste zu öffnen. Sobald Sie den Anfang des gesuchten Begriffs eingeben, rollt die Liste an die betreffende Stelle.

PROZEDUREN

Der Button

Den Button findet man in der Komponentenpalette Standard.



Klicken Sie das Symbol an und ziehen Sie anschließend mit der Maus im Formular ein mehr oder weniger großes Rechteck auf: Die Anschrift ändern Sie im Objektinspektor unter 'Caption'. Doppelklick in den Button schreibt die Rahmenbedingungen der Prozedur ins Editorfenster.

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


end;
```

Ein Tipp: mehrere gleiche Button erhält man mit Kopieren und Einfügen, gleichmässig ausrichten lassen sie sich mit 'Ausrichten' im Kontextmenü.

Ereignisse, die zum Formular gehören: OnCreate

Mit Doppelklick ins Formular erhalten Sie:

```
procedure TForm1.FormCreate(Sender: TObject);
begin

end;
```

Fügen Sie hier Befehle ein wie:

Randomize;	(initialisiert den Zufallszahlgenerator)
WindowState:=wsMaximized;	(Setzt des Ausführungsfensters auf volle Bildschirmgrösse)
Form1.Height:=600;	
Form1.Width:=800;	(legen die Grösse des Ausführungsfensters fest)
Form1.Left:=20;	
Form1.Top:=0;	(legen die Lage des Ausführungsfensters fest)
Form1.Color:=RGB(255,215,0);	(legt die Farbe des Ausführungsfensters fest)

Die Befehle in diesem Ereignis werden ausgeführt, wenn Sie die Exedatei starten.

Weitere Ereignisse, die zum Formular gehören

Andere zum Formular gehörende Ereignisse erhalten Sie, wenn Sie im Objektinspektor die Karte 'Ereignisse' wählen, und das weiße Feld neben der Bezeichnung doppelklicken.

OnClick

```
procedure TForm1.FormClick(Sender:TObject);
begin

end;
```

Das Ereignis wird mit einem Klick ins Ausführungsfenster gestartet.

OnDblClick

```
procedure TForm1.FormDblClick(Sender: TObject);
begin

end;
```

Schreiben Sie hier hinein zum Beispiel ein: Close

Im Exe–Modus genügt ein Doppelklick ins Ausführungsfenster, um die Prozedur zu beenden.

OnPaint

```
procedure TForm1.FormPaint(Sender:TObject);
begin

end;
```

Diese Prozedur eignet sich vorzüglich, um eine Grafik mit Start der Exedatei zeichnen zu lassen.

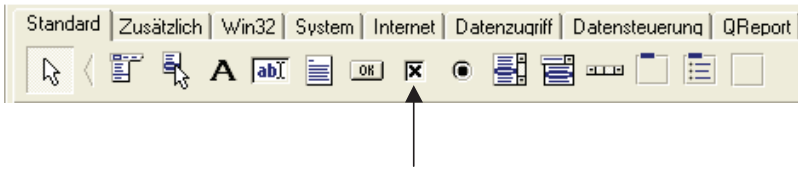
Ergänzung

Auch zu Buttons, Labels und Editfeldern gibt es eine ganze Reihe von Ereignissen;

wenn Sie z. B. ein Editfeld mit dem Ereignis 'OnChange' belegen, lässt sich eine Berechnung allein dadurch ausführen, dass Sie eine neue Zahl ins Editfeld tippen.

Die Checkbox

Die Checkbox finden Sie in der Komponentenpalette Standard.

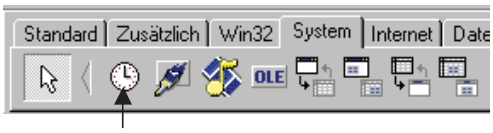



```
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
  while Checkbox1.Checked do
   begin
        . . .
        Application.ProcessMessages;
   end;
end;
```

Die Prozedur läuft so lange, wie die Checkbox angekreuzt ist.
Eine Checkbox wirkt auf dem Bildschirm viel unaufdringlicher als ein Button.

Der Timer

Den Timer finden Sie in der Komponentenpalette System:



Der Timer hat eine einzige Eigenschaft: Intervall
Sie beschreibt, in welchen Abständen ein Ereignis stattfinden soll. (1000 = 1 Sekunde)

Doppelklick des Timers führt auf:

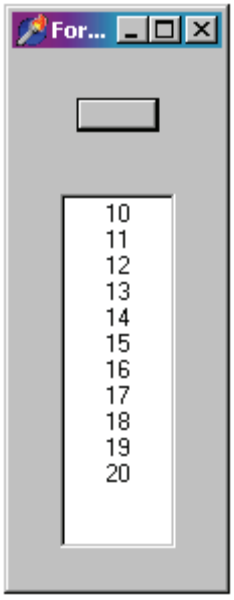
```
procedure TForm1.Timer1Timer(Sender: TObject);
begin

end;
```

Beispiele: Intervall:=5000 Close (Das Formular schliesst nach 5 Sekunden wieder)
Intervall:=1000 Label1.Caption:=TimeToStr(Time); (zeigt im Sekundentakt die Zeit an.
Grafikanimationen (Übung Lichterspiel)

Um den Timer zu beenden (ohne das Fenster zu schliessen), müssen Sie eine Abbruchbedingung einbauen:
Falls Sie in der Prozedur einen Zähler z eingebaut haben, könnte sie heissen:

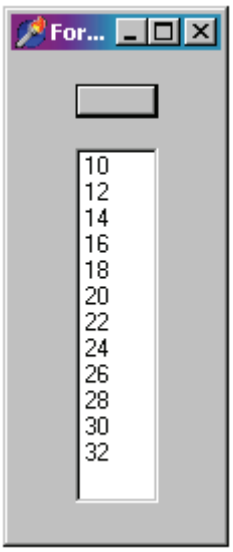
```
if z=100 then Timer1.Enabled:=false;
```

Schleifen



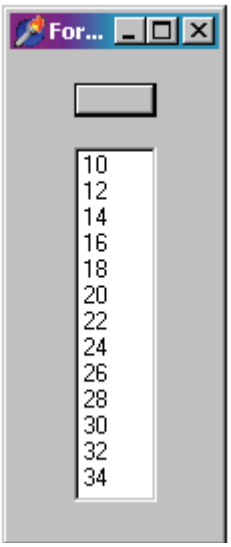

```
procedure TForm1.Button1Click(Sender: TObject);
begin
    for i:=10 to 20 do
        Memo1.Lines.Add(IntToStr(i));
end;
```

*Mehrere Anweisungen müssen mit **begin** und **end** eingefasst werden.*




```
procedure TForm1.Button1Click(Sender: TObject);
begin
    i:=10;
    while i<34 do
        begin
            Listbox1.Items.Append(IntToStr(i));
            i:=i+2;
        end;
end;
```

Zuerst wird die Bedingung geprüft, dann werden die Aktionen ausgeführt.




```
procedure TForm1.Button1Click(Sender: TObject);
begin
    i:=10;
```

Repeat

```
        Listbox1.Items.Append(IntToStr(i));
        i:=i+2;
    until i>35
end;
```

Die Bedingung wird erst geprüft, nachdem die Aktionen ausgeführt wurden.

Verzweigungen und Sprünge

If . . . then . . .else . . .;

Beispiele:

```
if a>=30 then Note:='Ausgezeichnet!';
```

```
if a>23  then Ergebnis:='Bestanden!'
         else Ergebnis:='nicht bestanden!';
```

```
if a>b then
          begin
             c:=a;
             a:=b;
             b:=c;
          end;
```

```
if a>b  then
          begin
             c:=a;
             a:=b;
             b:=c;
          end
       else
          begin
             a:=a-10;
             b:=b+10;
          end;
```

Beachten Sie:

Hinter dem **if** muß eine Aussage stehen, die man mit wahr oder falsch beantworten kann. meistens wird das eine Ungleichung oder eine Gleichung sein.

Mehrere Anweisungen nach dem **then** bzw. **else** müssen mit begin und end zusammengefaßt werden.

Beachten Sie die Semikolon!

Verzweigungen, wie die if-Anweisung oder die Case-Anweisung der nächsten Seite, sowie alle Schleifentypen können beliebig ineinander verschachtelt werden!
Achten Sie durch sorgfältige Darstellung (Einrücken!) darauf, dass die Übersicht gewahrt bleibt!

CASE . . . OF . . . (ELSE) . . . END;

Mit dieser Verzweigungsanweisung lassen sich oft eine Reihe aufeinanderfolgender oder ineinander verschachtelter **if** – Anweisungen vermeiden. Vorbedingung ist allerdings, daß der Ausdruck in der Case-Anweisung von ordinalem Typ ist; d. h. man muss ihn für eine Aufzählung brauchen können, bzw. er muss vom Typ Integer oder Char sein.

Beispiel: c sei eine Variable des Typs Char.

```
Case c of
    'A':   s:='Sie haben ein grosses A eingegeben!';
    'a':   s:='Sie haben ein kleines a eingegeben!';
    'B':   s:='Sie haben ein grosses B eingegeben!';
    'b':   s:='Sie haben ein kleines b eingegeben!';
else
          s:='Sie haben ein anderes Zeichen eingegeben!';
end;
```

Mit **if** könnte dasselbe Programm wie folgt aussehen:

```
if c='A' then  s:='Sie haben ein grosses A eingegeben!';
if c='a' then  s:='Sie haben ein kleines a eingegeben!';
if c='B' then  s:='Sie haben ein grosses B eingegeben!';
if c='b' then  s:='Sie haben ein kleines b eingegeben!';
         else  s:='Sie haben ein anderes Zeichen eingegeben!';
```

Besonders vorteilhaft ist bei der Anwendung von Case, dass man sehr einfach Bereiche benützen kann:

Beispiel: Punktzahl sei eine Variable des Typs Integer.

```
Case Punktzahl of
    0.. 9:             Zensur:='schwache Leistung';
   10..19:             Zensur:='ungenügend';
   20..29:             Zensur:='genügend';
   30..39:             Zensur:='gut';
   40..45:             Zensur:='sehr gut';
  -10..-1, 46..100:    Zensur:='diese Punktzahlen existieren gar nicht!':
end;
```

GOTO, CONTINUE, BREAK UND EXIT

Diese Bedingungen sollten in Delphi möglichst vermieden werden.

Funktionen

Funktionen kennen Sie schon aus der Mathematik:
sin(30°) gibt einen Wert zurück, nämlich 0.5. sin(30°) können Sie verwenden wie irgend eine Variable.

z. B. a:=sin(π/6);
b:= sin(π/6)+5;
Edit1.Text:=IntToStr(a);

Sie könnten aber die 1. und die 3. Anweisung zusammenfassen, und dabei eine Variable a sparen.

Edit1.Text:= IntToStr (sin(π/6));

Auch in Delphi kennen Sie schon eine ganze Reihe von vorgegebenen Variablen:

z. B. Arithmetische Funktionen: Abs(x), Int(x), sin(x), cos(x), Sqr(x), Sqrt(x), ArcTan(x), Random(x), Pi

Stringfunktionen: Length(x), IntToStr(x), Delete(wort, 2,1), LowerCase(x) und viele andere

Datums- und Zeitfunktionen: Date, DayOfWeek(datum)

Wie Sie sehen, gibt es Funktionen ohne Argument (Pi, Randomize), mit einem Argument (cos(x)) oder mit mehreren Argumenten (Delete, Insert).

Funktionen können wir auch selber definieren; das lohnt sich immer dann, wenn wir sie innerhalb einer Prozedur mehrmals benötigen, vielleicht auch, wenn wir eine Prozedur übersichtlicher gestalten möchten.

So sieht eine Funktion aus:

```
function Vzyl(r,h:Double):Double;
begin
   result:= pi*r*r*h;
end;
```

```
function Halb(a:Integer):Double;
begin
   Halb:=a/2;
end;
```

Zwischen **begin** und **end** können natürlich auch viel mehr Anweisungen stehen.

Sie sehen hier übrigens, dass in Funktionen eine Variable "result" benützt werden kann, die nicht definiert werden muss. Sie übernimmt in diesem Beispiel die Stelle von Vzyl

Aufgerufen (im Rahmen einer Prozedur) werden diese Funktionen z. B. mit:

```
Vzyl(3.75,1.8)
Halb(5)
```

Funktionen können nach **implementation**
{$R *.DFM}

eingefügt werden. Dann können sie von allen folgenden Prozeduren benützt werden.

Wenn Sie sie nur in einer einzigen Prozedur benützen, können Sie den Quelltext für die Funktion auch zwischen **procedure** und **begin** einfügen.

In "**Function** Halb(a:Integer):Double;" ist:

a ein Übergabeparameter vom Typ Integer.
a wird innerhalb der Funktionsdeklaration benützt.

Der Rückgabeparameter der Funktion, das wäre bei Halb(5) die Zahl 2.5, ist vom Typ Double.

Aufrufen lässt sich die Funktion mit jedem beliebigen Parameter:

```
b:=StrToInt(Edit1.Text);
Edit2.Text;=IntToStr(Halb(b));
```

Mehrere Variabeln können durch Komma getrennt gleichzeitig definiert werden, wenn Sie vom gleichen Typ sind. Variablen verschiedenen Typs trennt man durch Strichpunkt:

```
function Vzyl(r,h:Double):Double;
function Hoch(a:Real; n: Integer):Real;
```

(Hoch(2.5, 2) ergibt $2.5^2 = 6.25$)

Übrigens:

Sie können in einer Prozedur auch andere Prozeduren aufrufen, die weiter oben definiert wurden.
Das ist ein Mittel, um komplizierte Strukturen übersichtlicher zu gestalten, indem man Teile davon in separate Prozeduren auslagert.

Eine Prozedur kann sich sogar selber wieder aufrufen, was aber mit Vorsicht zu bewerkstelligen ist, da das sehr leicht zu einer Endlosschleife führen kann. Dasselbe gilt für Funktionen.

Ein schönes Beispiel dafür ist die Funktion zur Berechnung einer Fakultät:

$n! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1$
z. B. $5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$

```
function fak(n:Integer):Integer;
begin
    if n=1 then result:=1 else result:=fak(n-1)*n;
end;
```

Grafik

In Delphi gibt es grundsätzlich drei Varianten um Grafik anzuzeigen:

- Anzeigen von fertigen Bildern mit Image oder PaintBox
- Zeichnen von Figuren mit Shape
- Einsatz von Grafikmethoden (d. h. Programmierung) auf dem Canvas (engl. Leinwand)

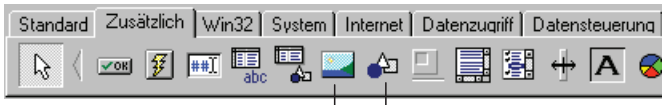


Image Shape

PaintBox

Shape

Mit Shape können Sie Formen wie Rechteck, Ellipsen, Kreise auf die gleiche Art wie z. B. ein Button oder ein Label erzeugen.

Im Objektinspektor können Sie die Figur gestalten:

- Shape Form auslesen: Rechteck und Quadrat (auch mit abgerundeten Ecken)
  Ellipse und Kreis

- Brush Füllmuster und Füllfarbe;
  Damit diese Optionen zum Vorschein kommen, müssen sie das + vor Brush doppelklicken.
  Unter Color finden Sie die Farbe, unter Style diverse Füllmuster.

- Pen Gestaltung der Randlinie;
  + doppelklicken für die Optionen Color, Style (Strichmuster), Width (Strichbreite in Punkt).
  Achtung! nur ausgezogene Linien (psSolid) können breiter als 1 gezeichnet werden.

Muster dazu auf den nächsten Seiten!

Auf den folgenden Seiten geht es nun vor allem darum, Grafiken mittels Programmierung herzustellen.

TCanvas dient als Zeichenfläche für Objekte, die sich selbst zeichen. Die unter Windows üblichen Steuerelemente wie z.B. Eingabe- oder Listenfelder benötigen keine Zeichenfläche, da sie von Windows gezeichnet werden.
TCanvas stellt Eigenschaften, Ereignisse und Methoden zur Verfügung, die beim Erzeugen von Bildern hilfreich sind, indem sie

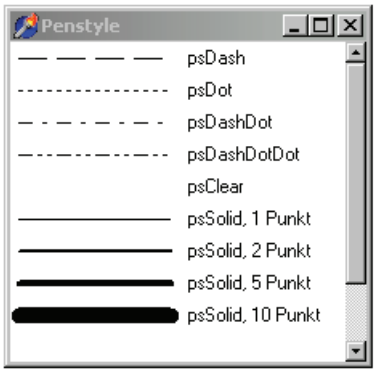
- die Art des verwendeten Pinsels und Stiftes sowie die Schriftart festlegen.
- eine Vielzahl von Fomen und Linien zeichnen und füllen.
- Text ausgeben.
- Grafiken zeichnen.
- eine Reaktion auf Änderungen am aktuellen Bild ermöglichen.

Muster zu Pen

Der Befehl heisst:

```
Canvas.Pen.Style:=psDash;
```

Der gewählte Style gilt für Linien und Ränder von Figuren; er bleibt eingestellt, bis er durch einen neuen Style-Befehl ersetzt wird.

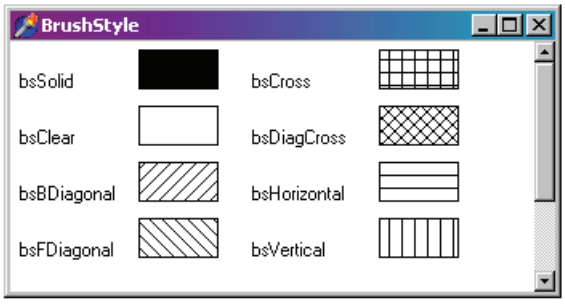


Muster zu Brush

Der Befehl heisst:

```
Canvas.Brush.Style:=psSolid;
```

Die Musterlinien werden in der aktuellen Musterfarbe auf dem bestehenden Hintergrund gezeichnet. Der Rand kann mit Pen.Color oder Pen.Style unabhängig davon gezeichnet werden. Der gewählte Style bleibt eingestellt, bis er durch einen neuen Style-Befehl ersetzt wird.



Farben

Systemfarben	clAqua	Aqua (Blaugrün)	clMaroon	Maroon (Kastanienbraun)
	clBlack	Schwarz	clNavy	Marineblau
	clBlue	Blau	clOlive	Olivgrün
	clDkGray	Dunkelgrün	clPurple	Purpur
	clFuchsia	Fuchsia (rosa)	clRed	Rot
	clGray	Grau	clSilver	Silber
	clGreen	Grün	clTeal	Teal (dunkles Blaugrau)
	clLime	Lime (mittleres Grün)	clWhite	Weiß
	clLtGray	Hellgrün	clYellow	Gelb

Systemfarben werden folgendermassen zugewiesen:

```
Canvas.Pen.Color:=clSilver;
Canvas.Brush.Color:= clBlue;
```

Pen- und Brush-Farben können unabhängig voneinander gewählt werden.
Sie bleiben eingestellt, bis sie durch einen neuen Farbbefehl aufgehoben werden.

RGB-Farben

Die Farben werden durch Zahlen dargestellt, die die Anteile an Rot, Grün, Blau wiedergeben und zwar üblicherweise in Hexadezimalzahlen.

z. B.
```
Canvas.Brush.Color:=$000000FF    (rot)
Canvas.Brush.Color:=$0000FF00    (grün)
Canvas.Brush.Color:=$00FF0000    (blau)
Canvas.Brush.Color:=$00000000    (schwarz)
Canvas.Brush.Color:=$00FFFFFF    (weiss)
```

FF ist die grösste zweistellige Zahl im Hexadezimalsystem, das die 16 Ziffern 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F aufweist und entspricht im Dezimalsystem der Zahl 255.

00	BF	7A	CD
	Blau	Grün	Rot

BF: $11 \cdot 16 + 15 = 191$ Teile Blau
7A: $7 \cdot 16 + 10 = 122$ Teile Grün
CD: $12 \cdot 16 + 13 = 205$ Teile Rot

Ohne Hexadezimalzahlen lässt sich die Farbe $00BF7ACD zuweisen als:

```
Canvas.Pen.Color:=RGB(191,122,205)
```

Mit den Extremwerten:

```
Canvas.Pen.Color:=RGB(0,0,0)          (Schwarz)
Canvas.Pen.Color:=RGB(255,255,255)    (Weiss)
```

Koordinaten

Das Koordinatensystem beruht auf der Einheit Punkt oder Pixel. Der Nullpunkt ist in der linken oberen Ecke des Formulars.

Sie sehen im Objektinspektor wie breit (Width) und wie hoch (Hight) das Formular momentan ist.
Falls Sie die Grösse des Formulars noch verändern, arbeiten Sie besser mit den Zahlen CustomWidth und CustomHight.

(CustomWidth-20, CustomHight-10) sind die Koordinaten eines Punktes, der vom rechten Rand 20 Pixel und vom untern Rand 10 Pixel Abstand hat.

Der Punkt mit den Koordinaten (CustomWidth **div** 2, CustomHight **div** 2) liegt etwa in der Mitte des Formulars. Beachten Sie das **div**! Koordinaten müssen ganzzahlig sein!

Tipp: Wenn Sie sich die Koordinaten nicht ganz vorstelllen können:
zeichnen Sie ein Label und lesen Sie die Zahlen im Objektinspektor ab.
(Left,Top) ergibt die Ecke oben links, (Left+Width,Top+Hight) die Ecke unten rechts.

Figuren

Punkt

Auf einen Punkt greifen Sie mit der Eigenschaft Pixels zu:

z. B. den Punkt (50,20) schwarz färben: Canvas.Pixels[50,20]:=clBlack

Da Pixels eine Eigenschaft ist, können Sie damit auch die Farbe eines Punktes ablesen:

```
var
  farbe: Tcolor;
farbe:=Canvas.Pixels[50,20];
```

Um abzulesen, wo sich der Stift im Moment befindet, benützen Sie die Eigenschaft PenPos:

```
var
  Ecke: Tpoint;
  xwert: LongInt;
Ecke:=Canvas.Pos;
xwert:=Canvas.Pos.X;
```

Geraden

Um Geraden zu zeichnen benötigen Sie zwei Befehle;

```
Canvas.MoveTo(x1,y1);
Canvas.LineTo(x2,y2);
```

Zahlenbeispiel:

```
Canvas.MoveTo(70,50);
Canvas.LineTo(410,120);
```

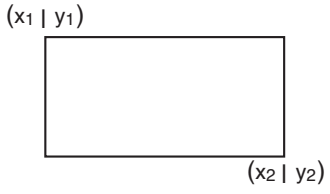
Es ginge auch mit der Methode PolyLine:

```
Canvas.PolyLine([Point(x1,y1),Point(x2,y2)]);
```

Rechteck und Quadrat

Linke obere und rechte untere Grenze angeben:

```
Canvas.Rectangle(x1,y1,x2,y2);
```



Zahlenbeispiel:

```
Canvas.Rectangle(70,50,170,100);
```

Ein Quadrat hat gleiche Seitenlängen:

```
Canvas.Rectangle(70,50,100,80);
```

$$\begin{pmatrix} 70+30=100 \\ 50+30=100 \end{pmatrix}$$

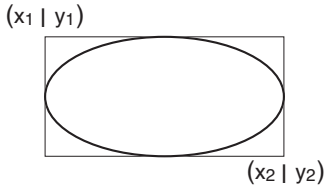
Ellipse und Kreis

Linke obere und rechte untere Grenze des Rechtecks angeben:

Canvas.Ellipse(x_1,y_1,x_2,y_2);

Zahlenbeispiel:

Canvas.Ellipse(70,50,170,100);



Ein Kreis mit Mittelpunkt (u,v) und Radius r:

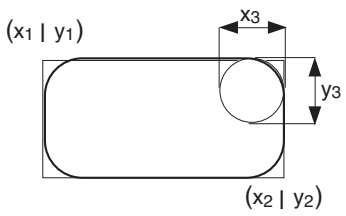
Canvas.Ellipse(u-r,v-r,u+r,v+r);

Abgerundetes Rechteck

Canvas.RoundRect(x_1,y_1,x_2,y_2,x_3,y_3);

Zahlenbeispiel:

Canvas. RoundRect(70,50, 170,100,20,20);



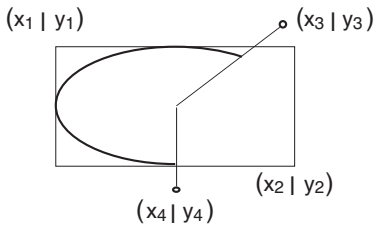
Bogenstück

Das Bogenstück kann nicht gefüllt werden.

Canvas.Arc(x_1,y_1,x_2,y_2,x_3,y_3,x_4, y_4);

Zahlenbeispiel:

Canvas. Arc(70,50, 170,100, 150,50, 120,100);

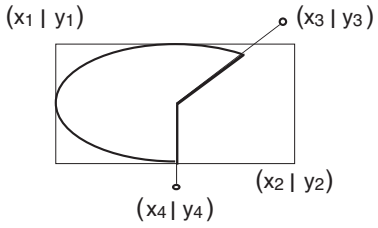


Tortenstück

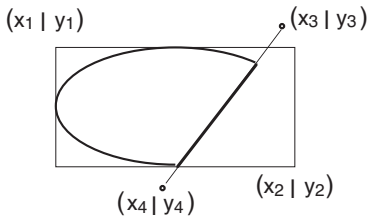
Canvas.Pie(x_1,y_1,x_2,y_2,x_3,y_3,x_4, y_4);

Zahlenbeispiel:

Canvas. Pie(70,50, 170,100, 150,50, 120,100);



Kreissegment



Canvas.Cord(x1,y1,x2,y2,x3,y3,x4, y4);

Zahlenbeispiel:

Canvas. Cord(70,50, 170,100, 150,50, 120,100);

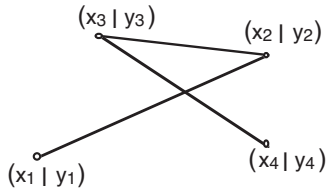
Streckenzug

Der Streckenzug kann nicht gefüllt werden.

Canvas.PolyLine([Point(x1,y1),Point(x2,y2), Point(x3,y3), Point(x4, y4), . . .]);

Zahlenbeispiel:

Canvas.PolyLine([Point(5,22), Point(37,9), Point(14,4), Point(37,19), . . .]);



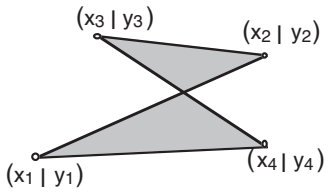
Vieleck

Das Polygon schliesst sich selbst und kann gefüllt werden.

Canvas.Polygon([Point(x1,y1),Point(x2,y2), Point(x3,y3), Point(x4, y4), . . .]);

Zahlenbeispiel:

Canvas.Polygon([Point(5,22), Point(37,9), Point(14,4), Point(37,19), . . .]);



TEXTAUSGABE

Einfachste Methode um Text auf den Bildschirm zu bringen.

Als Schriftart wird der aktuelle Wert von Font verwendet.
Zur Bezeichnung der Lage geben Sie die Koordinaten der Ecke oben links an.

Beispiel:

```
Canvas.Font.Height:=20;
Canvas.TextOut(20,10,'HALLO!');
```

Etwas Theorie zu Strings

Ein String kann bis zu 2 Giga einzelne Zeichen (Buchstaben, Ziffern, Sonderzeichen) umfassen, also praktisch unbegrenzt lang sein.
(2 Giga $= 2 \cdot 10^9$, ungefähr 1.5 Millionen Schreibmaschinenseiten!)

Einzelne Zeichen werden als Variablen des Typs **Char** definiert.

Eine Auswahl von Befehlen rund um Strings:

Es seien folgende Variablen deklariert:

```
Vorname:='Hanspeter';
Nachname:='Muster';
Titel:='Delphi';
```

dann ergeben die folgenden Befehle:

Titel+'buch'	→	Delphibuch
Vorname+" "+Nachname	→	Hanspeter Muster
Copy(Vorname,4,2)	→	sp
Vorname[4]	→	s
Delete(Vorname,4,2)	→	Haneter
Insert('-',Vorname, 5)	→	Hans-peter
Length(Vorname)	→	9
LowerCase(Titel)	→	delphi
UpperCase(Titel)	→	DELPHI

IntToStr, IntToFloat und ihre Umkehrung sind längstens bekannt.

Zu jedem Zeichen gehört eine Zahl (0 bis 255 vom Typ Byte), bekannt unter dem Namen ASCII-Code (American Standard Code for Information Interchange).

Ord('A')	→	65	Char(65)	→	A
Ord('B')	→	66	Char(66)	→	B
Ord('Z')	→	90	Char(90)	→	Z
Ord('a')	→	97	Char(97)	→	a
Ord('z')	→	122	Char(122)	→	z

StringGrid

Im Buch befassen sich die Seiten 343/4 und 347 damit, allerdings in einer nicht selbständig zu benützender Anleitung.

In der Werkzeugleiste "Zusätzlich" finden Sie die StringGrid, die Sie wie ein Rechteck aufziehen.

Viele **Eigenschaften** finden Sie im Objektinspektor; Erklärungen dazu im Hilfe-Index unter dem Stichwort "TStringGrid".

StringGrid1.RowCount	gibt die Anzahl der Zeilen an
StringGrid1.ColCount	gibt die Anzahl der Spalten an
StringGrid1.DefaultColWidth	Spaltenbreite
StringGrid1.DefaultRowHeight	Zeilenhöhe
StringGrid1.ScrollBar	Art der Rollbalken
StringGrid1.FixedCols	eine (oder mehr) Spalten am linken Rand, die immer sichtbar bleiben.
StringGrid1.FixedRows	der entsprechende Befehl für den Tabellenkopf

StringGrid1.Options (doppelklicken!):

goEditing	bestimmt, ob man in der Tabelle bei laufender Anwendung Einträge machen kann oder nicht.
goTabs	true: man kann die Tabulator-Taste benützen, um in die nächste Zelle zu kommen

Die Zeilen bzw. die Spalten werden fortlaufend nummeriert, beginnend bei **0**.

Auf eine Zelle greift man zu mit:

StringGrid1.Cells[3,7] (das wäre, da man von 0 an zählt, die 4. Zelle in der 8. Zeile)

StringGrid1.Cells[0,Row] ist die 1. Zelle in der aktuellen Zeile.

a:=StrToInt(StringGrid1.Cells[2,3]); weist a den Eintrag der Zelle als Zahl zu.

Oft greift man mit Schleifen auf die Zellen zu:

```
For i:=0 to 5 do StringGrid1.Cells[0,i]:=i+1;
```

(Schreibt in der vordersten Spalte von oben nach unten die Zahlen 1 2 3 4 5 6)

Sie können darauf verzichten, Dutzende von Malen StringGrid1 zu schreiben.
Verpacken Sie die Befehle in:

```
With StringGrid1 do
  begin

  end;
```

Ausserdem:

Spaltenbreiten und Zeilenhöhen können Sie auch auf dem Formular von Hand verändern; verschieben Sie dazu mit der Maus die Begrenzungslinie zwischen zwei Zellen einer fixierten Zeile oder Spalte.

Eigenschaften können Sie – wie gewohnt – auch im Programmcode zuweisen:

StringGrid1. ColWidths[0]:=25; setzt die Breite der 1. Spalte auf 25.
StringGrid1.Options:=[goEditing, goTabs] setzt die genannten Eigenschaften auf true.
StringGrid1.RowCount gibt die Zeilenzahl an.

Die String-Liste

Eine Stringliste ist eine Liste von Wörtern. Ein "Wort" besteht aus dem Text einer Zeile, z. B. come, came, come
In meinen Beispielen trägt die Stringliste den Namen "woerter".

Erste Schritte

1. Schritt: Im **type**-Teil des Quelltextes finden Sie die Rubrik **puplic**.
Geben Sie darunter den (oder die) Namen Ihrer Listen an:

```
type
  woerter: Tstrings
```

2. Schritt: Erzeugen Sie die Liste in der FormCreate-Prozedur:

```
woerter:=TstringList.Create;
```

Arbeiten mit Stringlisten (Auswahl)

Wörter aus einer Textdatei (L109.txt) in die Stringliste einlesen:

```
woerter.LoadFromFile('L109.txt');
```

Die Stringliste in einer Textdatei speichern:

```
woerter.SaveToFile('R109.txt');
```

Die Liste in einem Memo (oder einer Listbox oder Combox) anzeigen:

```
Memo1.Lines:=woerter;
```

Das 3. Wort der Liste anzeigen (das Zählen beginnt bei Null!):

```
Edit1.Text:=woerter[2];
```

Das 5. Wort der Liste löschen:

```
woerter.delete(4);
```

Am Ende der Liste ein Wort anfügen:

```
woerter.add('view');
```

Anzahl n der Einträge der Stringliste feststellen:

```
n:=woerter.count;
```

Nota Bene: alle diese Befehle funktionieren auch mit Memo1.Lines bzw. mit ListBox1.Items

Melde- und andere Fenster

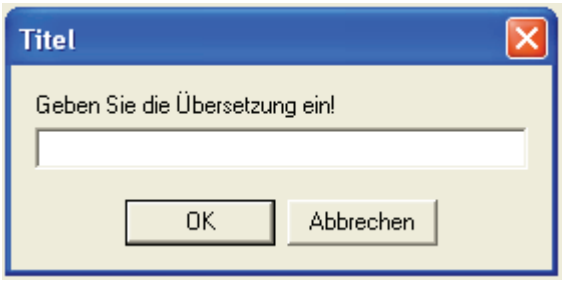
Meldefenster

ShowMessage('Das ist ein einfaches Messagefenster')

mit dem folgenden Befehl können Sie die Lage des Fensters angeben (alle andern Fenster erscheinen in der Mitte des Bildschirms):

ShowMessagePos('Text',100,30)

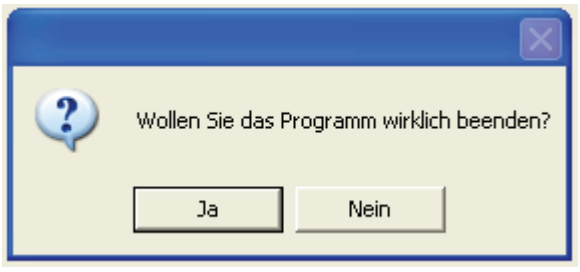
Eingabefenster



InputBox('Titel','Geben Sie die Übersetzung ein!',' ')

Sie können statt 'Titel' auch einfach einen Leerstring angeben ' '
In hintersten Leerstring könnte man einen Wert für das Eingabefeld vorgeben, der dann übernommen oder überschrieben werden kann.

Application.MessageBox



Die Titelleiste ist leer

Application.MessageBox('Wollen Sie das Programm wirklich beenden?',' ',36) gibt bei der Wahl von "Ja" den Wert 6, bei der Wahl von "Nein" den Wert 7 zurück.

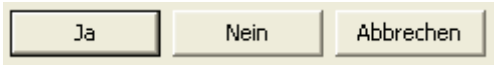
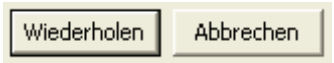
36=32+4+0 mehr dazu auf der Rückseite.

Die Application.MessageBox hat drei Parameter:

Application.MessageBox(Text,Caption,Flags)

Die Zahl Flags erhält man, indem man aus jeder der drei folgenden Gruppen eine Zahl ausliest und alle drei zusammenzählt.

								
0		16		32		48		64

	0	MB_OK
	1	MB_OKCANCEL
	2	MB_ABORTRETRYIGNOR
	3	MB_YESNOCANCEL
	4	MB_YESNO
	5	MB_RETRYCANCEL
	16384	MB_HELP

Die dritte Zahl gibt an, welche Schaltfläche den Focus hat, dass heisst, welche Schaltfläche mit der Enter-Taste betätigt werden kann:

erste Schaltfläche	0	MB_DEFBUTTON1
zweite Schaltfläche	256	MB_DEFBUTTON2
dritte Schaltfläche	512	MB_DEFBUTTON3

Rückgabewerte:

0	Fehler
1	Schaltfläche 'OK' gedrückt
2	Schaltfläche 'Abbrechen' gedrückt
3	Schaltfläche 'Abbrechen' gedrückt
4	Schaltfläche 'Wiederholen' gedrückt
5	Schaltfläche 'Ignorieren' gedrückt
6	Schaltfläche 'Ja' gedrückt
7	Schaltfläche 'Nein' gedrückt