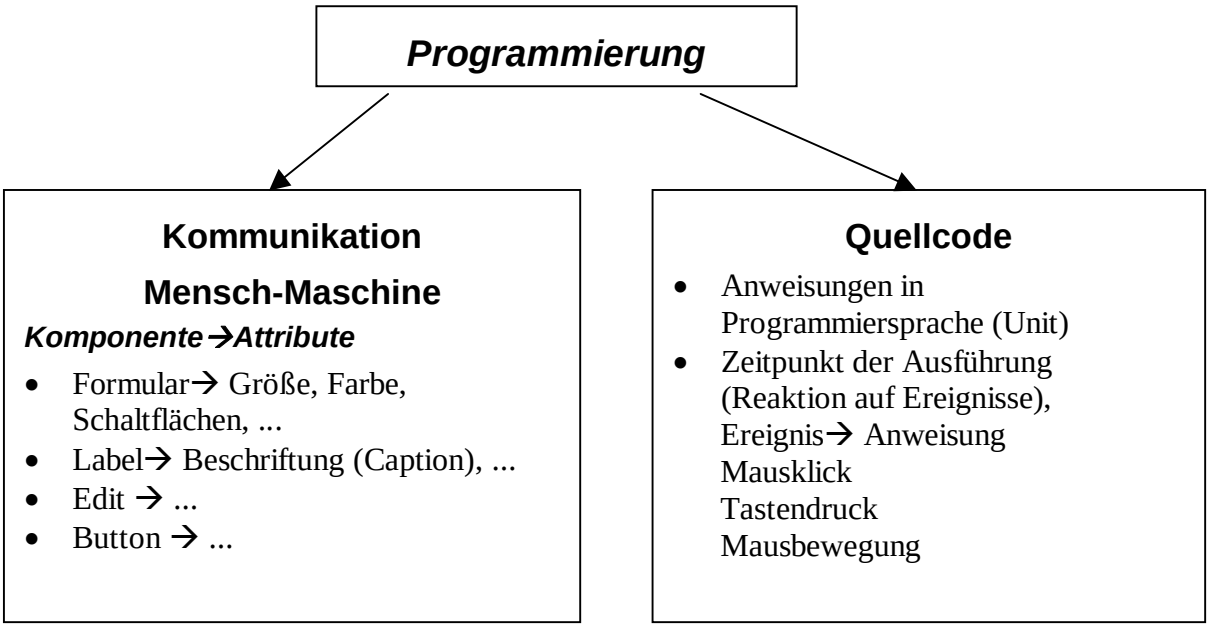
Programmierung mit Delphi



☝Für jedes Delphi-Programm einen Ordner anlegen!

Der Arbeitsbereich von Delphi

- Hauptfenster mit Menuleisten und Komponentenleiste
- Objekt - Explorer zum ändern der Attribute
- Das Formularfenster zum Gestalten der Programmoberfläche
- Das Editorfenster zur Eingabe des Quellcodes

Ändern von Eigenschaften mit dem Objekt-Explorer

Objekt auswählen | Attribut auswählen | Wert setzen

Reaktion auf Ereignisse

Komponente auswählen→Ereignis auswählen → Methode (Prozedur) festlegen
z.B. Button → Mausklick → Farbe des Formulars ändern

Ändern von Eigenschaften während der Programmausführung

Im Quellcode müssen die entsprechenden Anweisungen notiert werden.

```
Komponentenname.Eigenschaft := Wert;
Form1.left := 0;
Form1.color:= clred;
Button1.caption:= 'OK' ;
```

Dateien eines Delphi – Projektes

Dateierweiterung	Erläuterung
.pas	Quellcode (Unit)
.dfm	Formulardatei (Einstellungen des Formulars)

.dpr	Projektdatei
.dcu	in Maschinencode compilierte Unit
.exe	ausführbare Programmdatei

Aufbau einer Unit

Name der Unit

```
unit Unit1;

interface
```

Festlegen aller verwendeten Elemente(Komponenten, Methoden, Variable)

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

Verwendung anderer Units (z.B. für Standardkomponenten)

Typdefinition, z.B. Festlegen, welche Komponenten und Methoden das Formular enthalten bzw. verwenden soll

```
type
  TForm1 = class(TForm)
    Label1: TLabel;
   …
    procedure OKClick(Sender: TObject);
  private
    { Private-Deklarationen }
  public
    { Public-Deklarationen }
  end;

var
  Form1: TForm1;

Implementation
```

Anweisungsteil: Festlegen der Anweisungen für die verschiedenen Methoden

```
{$R *.DFM}

procedure TForm1.OKClick(Sender: TObject);
begin
 form1.color:= clred;
end;
```

Ende einer Methode (Prozedur)

```
end.
```

Ende der Unit

Übung Tag und Nacht

Neue Komponente: Image (Bildfeld)

Neue Eigenschaft: visible
Die Eigenschaft visible ist ein sogenannter Wahrheitswert (Boolean) mit den Werten true (wahr => sichtbar) und false (falsch => unsichtbar)

Neue Eigenschaft: Picture
Mit der Eigenschaft Picture wird die angezeigt Bilddatei festgelegt.

Sonnenaufgang - Die Timer – Komponente

Vorübung: Bei jedem Klick auf ein Button soll die Image-Komponente mit dem Sonnenbild weiter aufsteigen.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
image1.left:=image1.left + 10;
image1.top:=image1.top – 10;
end;
```

Die Timer-Komponente wiederholt eine Methode im festgelegten zeitlichen Abstand.

Eigenschaft	Bedeutung
Intervall	Zeit zwischen zwei Wiederholungen in Millisekunden
enabled	true: Timer ist aktiv, false: Timer ist deaktiviert

Bedingte Anweisung

Mit der If –Then Anweisung wird die Ausführung von Anweisungen an eine Bedingung gekoppelt.

```
If (Bedingung wahr) then Anweisung;

If (Bedingung wahr) then begin
                         Anweisung;
                         …
                         Anweisung;
                         end;
```

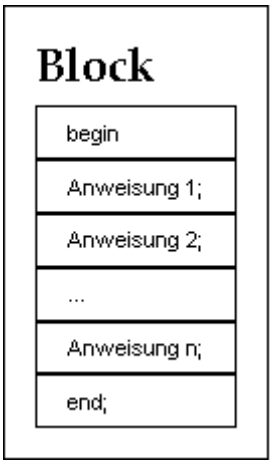
Anweisungs-block

Bsp.

```
If (form1.color = clBlack) then form1.color:= clBlue;
If (image1.visible = false) then image1.visible:=true;
If (timer1.enabled= false) then Begin
                                Timer1.interval:=100;
                                Timer1.enabled:=true;
                                End;
```

Gehören Anweisungen zusammen (z.B. zu einer Prozedur oder zu einer Bedingung), dann werden sie als Anweisungsblock zusammengefasst);

Struktogramm eines Anweisungsblocks



Übung Autofahrt

Mit Hilfe einer Timer-Komponente soll ein Auto von links nach rechts über das Formular fahren. Wenn das Auto das Formular rechts verlassen hat, soll es links neu erscheinen.

Übung Autorennen

Zwei Autos sollen gleichzeitig ein „Rennen“ fahren. Dazu wird werden beide in zufälligen Sprüngen über den Bildschirm bewegt.

Zufallszahlen

In Delphi können mit der random - Anweisung Zufallszahlen erzeugt werden. Dazu muss zunächst der Zufallsgenerator mit der Anweisung randomize gestartet werden.
Mit random(n) erzeugt man natürliche Zahlen zwischen 0 und n-1; d.h. mit der Anweisung random(4) wird eine der Zahlen 0, 1, 2, 3 ausgewählt.

```
Procedure form1.formcreate(Sender:Tobject);
Begin
Randomize; //Programmstart➔ Start des Zufallsgenerators
End;

Procedure form1.timer1timer(sender:Tobject);
Begin
ImAuto.left:= ImAuto.left + Random(10);
…
End;
```

Übung Ampel

Eine Ampel soll gesteuert werden.
Vorüberlegungen:

1. Wie viele und welche Ampelphasen gibt es? (rot, rot-gelb, grün, gelb)
2. Welche Komponenten werden benötigt?

Komponente	***Eigenschaften***	***Bedeutung***
timer1	name: timer1 intervall: 500 tag: 0	Steuerung der Ampel Länge der Phasen ungenutzt
button1	name:btStart caption: start	Ampel einschalten

button2	name:btStop caption: stop	Ampel ausschalten
shape1	name: shBox	Ampelkasten
shape2	name: shRed brush.color: clsilver	rote Ampel Licht aus
shape3	name: shYellow brush.color: clsilver	gelbe Ampel Licht aus
shape4	name: shGreen brush.color: clsilver	grüne Ampel Licht aus

3. Welche Aufgabe muss die Timer-Prozedur erfüllen?
   - Ampelphase feststellen
   - entsprechende Lampen ein- bzw. ausschalten
   - evtl. Intervall anpassen

Die unbenutzte Eigenschaft Tag des Timers wird benutzt um die Ampelphase festzulegen

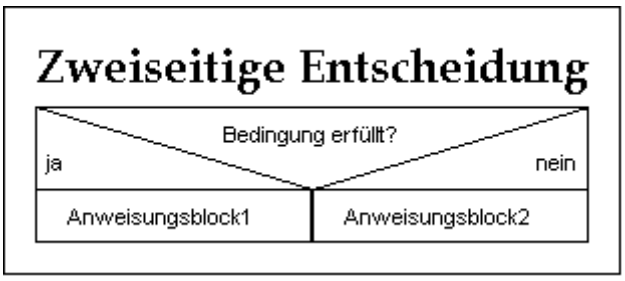
d.h. tag = 0 ➔ Rot, tag = 1 ➔ Rotgelb, tag = 2 ➔ Grün, tag = 3 ➔ Gelb

```
procedure Tform1.timer1OnTimer(Sender:Tobject);
begin
if timer1.tag = 0 then
   begin
   shRed.brush.color:= clred;          //Rot an
   shYellow.brush.color:= clSilver;    //Gelb aus
   shGreen.brush.color:= clSilver;     // Grün aus
   timer1.interval:= 500;              // Länge der Rotphase
   end;
if timer1.tag = 1 then
   begin
   shYellow.brush.color:= clYellow;    //Gelb an
   timer1.interval:= 100;         // Länge der RotGelbphase
   end;
…
timer1.tag:= timer1.tag + 1;     //Anpassen der Ampelphase
if Timer1.tag>3 then timer1.tag:=0;
end;
```

Zweiseitige Entscheidung

Bei Entscheidungen kann man zwischen zwei Alternativen auswählen:

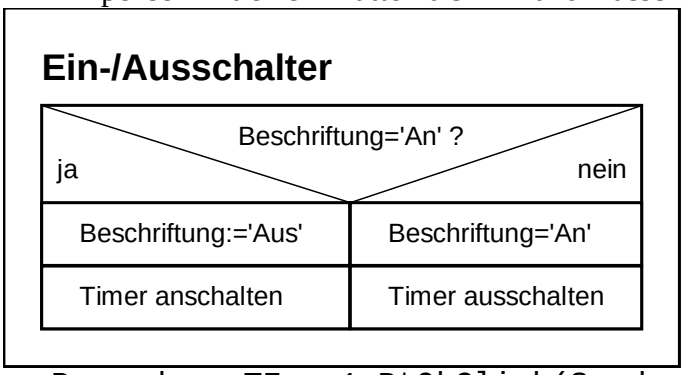
```
Wenn (das und das zutrifft)
  Dann mach dieses
  Sonst mach jenes;

If (Bedingung wahr)
  then Anweisung(sblock)
  else Anweisung(sblock);
```



Struktogramm einer zweiseitigen Entscheidung

Übung Ein/Aus-Schalter

Ein Ampel soll mit einem Button als Ein- und Ausschalter bedient werden.




```
Procedure TForm1.BtOkClick(Sender:Tobject);
Begin
If (BtOK.Caption=’An’) then
   Begin
   BtOK.Caption:=’Aus’;
   Timer1.enabled:=true;
   End
   Else
   Begin
   BtOK.Caption:=’An’;
   Timer1.enabled:=false;
   End;
End;
```

Übung Ampel mit Blinker-Modus

Eine Ampel soll nach dem Ausschalten Gelb blinken.

Lösung: 2. Timer einfügen, mit zwei „Ampelphasen“ (Gelb an – Gelb aus)

Übung Fußgängerampel

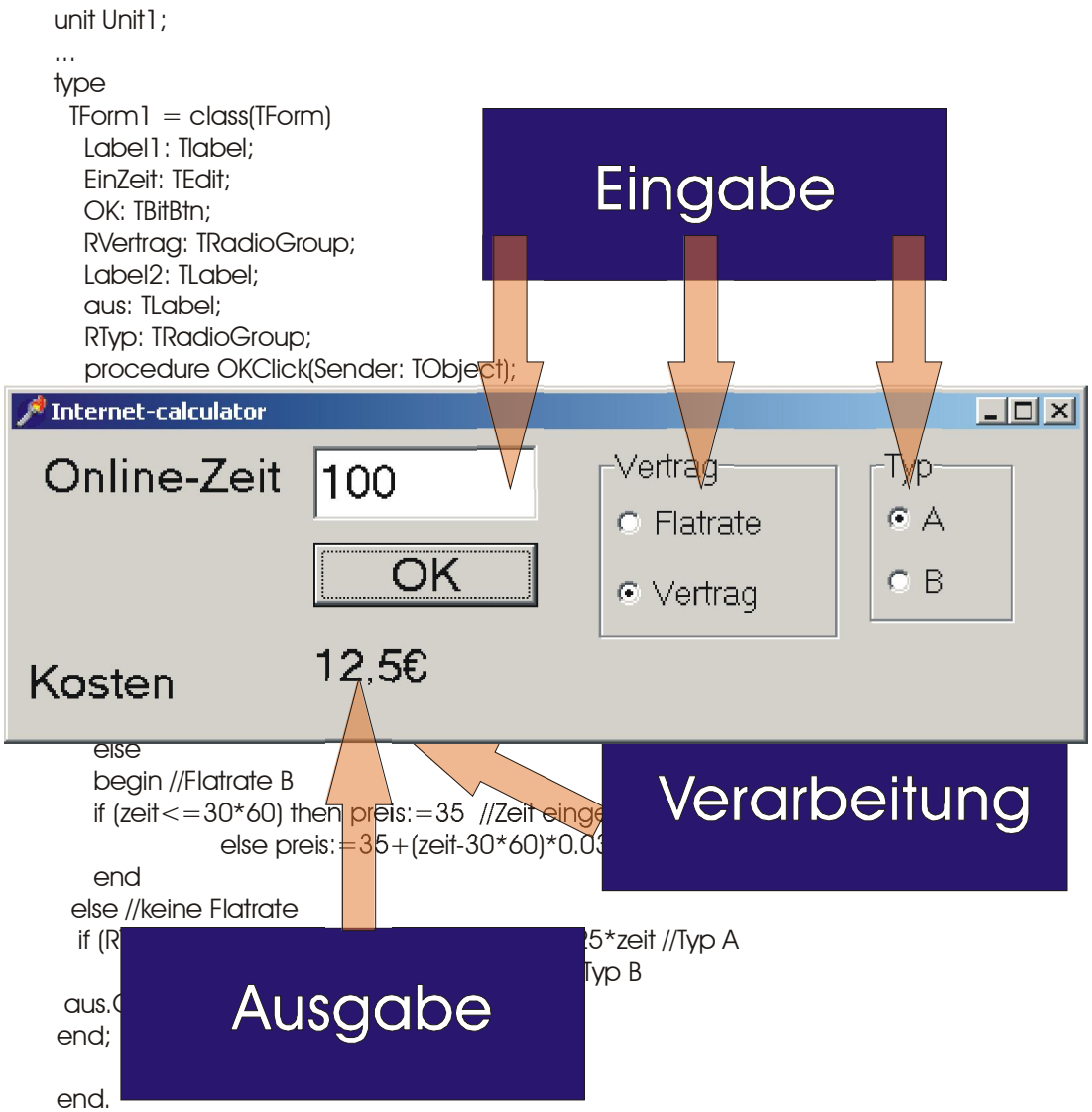
Mit Shapes soll eine Straße mit Fußgängerüberweg erzeugt werden, dazu eine Fahrzeugampel und eine Fußgängerampel, die sich wechselseitig in Betrieb setzen.

Lösung: Für jede Ampel einen Timer, das Umschalten erfolgt beim zurücksetzen der Tag-Eigenschaft

Das EVA – Prinzip

Ein Kalkulationsprogramm soll die Internetkosten eines Kunden berechnen, nachdem seine Online-Zeit sowie seine Vertragsbedingungen eingegeben wurden.

- Eingabe: Online-Zeit, Vertragsart,
- Verarbeitung: Berechnung der entstandenen Kosten
- Ausgabe: Anzeigen der berechneten Kosten



Eingabe und Ausgabe werden grundsätzlich über das Formular vorgenommen,
die Verarbeitung der eingegebenen Daten erfolgt in der CPU durch Ausführen der im Quellcode festgelegten Befehle.

Typische Komponenten zur Dateneingabe sind:
Textfelder (Edit), Radio- oder Checkboxen, Buttons

Typische Komponenten zur Datenausgabe sind:
Label, Messageboxen

Variable

Um Berechnungen durchführen zu können, benötigt man Variable. Jede Variable wird durch Angabe eines Bezeichners (Name) und ihre Variablentyps festgelegt, z.B.

```
Var Bezeichner : Variablentyp;
var Vorname : String; → Textvariable
var alter   : Integer; → Ganze Zahl
```

Beispiel: ein einfacher Taschenrechner

Komponente	*Eigenschaften*	*Bedeutung*
label1	name: label1 caption: 1. Zahl	Beschriftung
label2	name: label2 caption: 2. Zahl	Beschriftung
edit1	name: EZahl1 text: *leer*	1. Zahl eingeben
edit2	name: EZahl2 text: *leer*	2. Zahl eingeben
label3	name: Ausgabe caption: ---	Anzeigen des Ergebnisses
button1	name:btSumme caption: Summe	Summe berechnen

Eingaben und Ausgaben sind immer Texte(Strings)

d.h. Daten, die als Zahlen bearbeitet werden, müssen vom Texttyp `String` in einen Zahltyp (z.B. `Integer`) umgewandelt werden → Typkonvertierung.

```
procedure TForm1.btSummeClick(Sender:Tobject);
var zahl1, zahl2, summe : integer;
begin
zahl1:= StrToInt(EZahl1.text);
zahl2:= StrToInt(Ezahl2.text);

summe:= zahl1 + zahl2;

ausgabe.caption:= IntToStr(summe);
end;
```

Typkonvertierung der Eingabedaten

Berechnung

Typkonvertierung und Anzeige des Ergebnisses

Übung

Erweitere den Rechner auf die anderen Grundrechenarten!