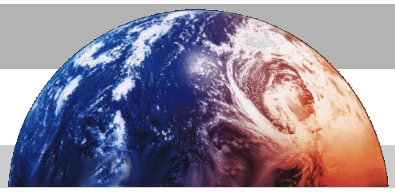
Lokale und globale Variablen



Lokale Variablen

In einer Prozedur wie der folgenden für einen Währungsumrechner, sind 2 Variablen (Dollarbetrag und Eurobetrag) deklariert, die bei den Berechnungen helfen.

```
procedure TForm1.BtnUmrechnenClick(Sender: TObject);
var
  Dollarbetrag,Eurobetrag: Real;
begin
  { Eingabe (Wert einlesen) }
  Dollarbetrag := StrToFloat(EdtDollar.Text);

  { Verarbeitung (Waehrung umrechnen) }
  Eurobetrag := Dollarbetrag / 1.4776;

  { Ausgabe }
  EdtEuro.Text := FloatToStrF(Eurobetrag,ffFixed,8,2);
end;
```

Lokale Variablen
(nur in dieser Prozedur gültig)

- Mit der Deklaration der Variablen „Dollarbetrag“ und „Eurobetrag“ im „var“-Teil der Prozedur wird im Hauptspeicher (auch genannt Arbeitsspeicher oder RAM) des Computers Speicherplatz für eben diese beiden Variablen reserviert. Sie stehen sodann in der Prozedur „BtnUmrechnenClick“ – und nur dort – zur Verfügung.
- Sobald DELPHI die Prozedur abgearbeitet und wieder verlassen hat, gibt Delphi den Speicherbereich der Variablen im Hauptspeicher für andere Aufgaben frei. Wir können dann *nicht* mehr auf die Variablenwerte zugreifen – sie gehen unwiderruflich verloren.
- Wird die Prozedur ein weiteres Mal aufgerufen, werden die Variablen „Dollarbetrag“ und „Eurobetrag“ erneut erzeugt. Diese haben dann aber *nichts* mit „Dollarbetrag“ und „Eurobetrag“ des vorigen Prozedurdurchlaufs zu tun – sie sind im Prinzip neue Variablen.
- Solche Variablen nennt man lokale Variablen, da sie nur lokal [= örtlich begrenzt] in der Prozedur gültig sind und *nicht* auch außerhalb dieser im restlichen Programm.
- Dem entsprechend müssen lokale Variablen in der entsprechenden Prozedur deklariert werden, und zwar zwischen den Schlüsselwörtern „procedure“ und „begin“.

Globale Variablen

Damit Variablen während der gesamten Laufzeit *überall* im Programm zur Verfügung stehen (d. h. in jeder Prozedur), müssen wir auf globale Variablen zurückgreifen. Betrachten wir dazu folgendes Beispiel:

```
unit uUnsinn1;

interface
```

```
uses
  Windows, Messages, SysUtils, [...];

type
  TForm1 = class(TForm)
    BtnZuweisen: TBitBtn;
    BtnAusgeben: TBitBtn;
    EdtAusgabe: TEdit;
    procedure BtnZuweisenClick(Sender: TObject);
    procedure BtnAusgebenClick(Sender: TObject);
  private
    { Private-Deklarationen }
    Wert: Integer;
  public
    { Public-Deklarationen }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.BtnZuweisenClick(Sender: TObject);
begin
  Wert := 42;
end;

procedure TForm1.BtnAusgebenClick(Sender: TObject);
begin
  EdtAusgabe.Text := IntToStr(Wert);
end;

end.
```

Globale Variable (in der gesamten Unit gültig, jedoch nicht in anderen Units – dazu müsste sie unter „public“ deklariert werden)

Wenn man auf den Button „BtnZuweisen“ klickt, wird der globalen Variablen „Wert“ der Wert 42 zugewiesen.

Wenn man auf den Button „BtnAusgeben“ klickt, wird die globale Variable „Wert“ in „EdtAusgabe“ ausgegeben.

- Erneut wird ein Bereich im Hauptspeicher reserviert, diesmal für die unter „private“ deklarierte Variable „Wert“. Im Gegensatz zu lokalen Variablen steht dieser in der gesamten DELPHI-Unit (jedoch nicht in anderen Projekt-Units, dazu dient „public“) zur Verfügung.
- Sobald DELPHI eine beliebige Prozedur abgearbeitet und verlassen hat, können wir *weiterhin* auf die globale Variable zugreifen – sie geht *nicht* verloren. Ihr Speicherplatz im Hauptspeicher wird erst mit Beendigung des Programms freigegeben.
- Somit gilt sie über die Prozedurgrenzen hinweg global [= umfassend, allgemein].
- Dem entsprechend dürfen globale Variablen nicht in einer bestimmten Prozedur deklariert werden, sondern weiter oben im allgemeinen „Interface“-Teil. Dabei gibt es theoretisch verschiedene Möglichkeiten. Wir wollen globale Variablen vorerst nur im Bereich „private“ deklarieren.

Fazit

- Es gilt der Grundsatz: lokal vor global.
  Variablen sollten wenn möglich immer *lokal* deklariert werden. Der Vorteil liegt darin,

dass sie nur während des jeweiligen Prozedurablaufs Speicherplatz im Hauptspeicher belegen und am Ende der Prozedur wieder freigeben, wenn die Variablen nicht mehr gebraucht werden. Außerdem kann es aus Gründen des Datenschutzes und der Datensicherheit sinnvoll sein, Daten – sprich: Variableninhalte – nur so lange im Speicher zu halten, wie erforderlich.
Umgekehrt sollten globale Variablen nur wenn nötig oder sinnvoll (z. B. weil sonst zu viel Programmierumstand) deklariert werden. Dies ist bspw. der Fall, wenn mehrere Prozeduren (oder Funktionen, mehr dazu später) auf ein und denselben Variableninhalt zugreifen müssen. Wären die Variablen dann nur lokal deklariert, würde ihr Inhalt nach Beendigung einer Prozedur verloren gehen – er könnte also nicht mehr in einer anderen Prozedur benutzt werden.
(Eine weitere Möglichkeit, auf globale Variablen zu verzichten, liegt der expliziten sog. „Variablenübergabe“ von einer Prozedur/Funktion an eine andere. Mehr dazu aber später.)

- Damit es keine Konflikte in der Bezeichnung von globalen und lokalen Variablen gibt, sollten stets unterschiedliche Namen verwendet werden. D. h. man sollte es vermeiden, dass bspw. eine globale Variable namens „Wert“ und eine weitere, lokale Variable „Wert“ existieren. Zwar ist so etwas prinzipiell erlaubt (dabei gibt Delphi stets der *lokalen* Variable den Vorrang vor der globalen!), doch sorgt es beim Programmieren für unnötige Verwirrung.

Aufgaben

Gegeben sei folgendes Übungsprogramm:

```
unit uTest;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs;

type
  TForm1 = class(TForm)
    BtnTueDieses: TButton;
    BtnTueJenes: TButton;
    procedure BtnTueDiesesClick(Sender: TObject);
    procedure BtnTueJenesClick(Sender: TObject);
  private
    { Private-Deklarationen }
    Zahl1,Zahl2: Integer;
    Wort: string;
  public
    { Public-Deklarationen }
  end;

var
  Form1: TForm1;
```

```
implementation

{$R *.dfm}

procedure TForm1.BtnTueDiesesClick(Sender: TObject);
var
  x,y: Real;
begin
  { Irgendwelche Anweisungen }
end;

procedure TForm1.BtnTueJenesClick(Sender: TObject);
var
  Str: string;
  Marke: Boolean;
begin
  { Irgendwelche Anweisungen }
end;

end.
```

1) Welche der obigen Variablen sind lokale, welche globale Variablen? Kreuzen Sie an.

	Zahl1	Zahl2	Wort	x	y	Str	Marke
lokale Variable							
globale Variable							

2) Welche der Variablen gilt in welcher Prozedur? Kreuzen Sie an.

	Zahl1	Zahl2	Wort	x	y	Str	Marke
in keiner (nirgends im Programm)							
nur in „BtnTueDiesesClick“							
nur in „BtnTueJenesClick“							
in beiden (im gesamten Programm)							