Modulares Programmieren[1]

Oftmals kommen in einem Programm bestimmte Anweisungen z.B. zur Eingabe, Ausgabe, Speichern, Drucken etc. öfter vor. Dann empfiehlt es sich, schon wegen der besseren Übersichtlichkeit, diese in eigenständige Teilprogramme (Module) auszulagern. Solche Module können dann auch unabhängig vom Programm immer wieder verwendet werden. Man unterscheidet drei verschiedene Arten von Modulen: *Funktionen, Prozeduren* und *Units*. Im Rahmen unseres Grundkurses wollen wir uns mit den Prozeduren und Funktionen beschäftigen.

Funktionen und Prozeduren sind demnach also eigenständige kleine in sich abgeschlossene Programmteile innerhalb eines Hauptprogramms. Man spricht deshalb auch von *Unterprogrammen*. Jede Funktion bzw. Prozedur besitzt einen Namen, über den sie von anderen Programmteilen aus aufgerufen werden können.

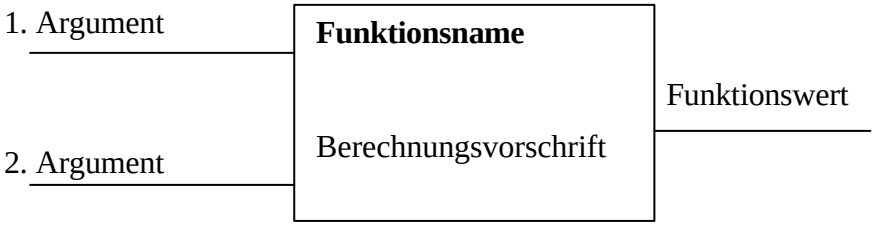
Der Unterschied zwischen Funktionen und Prozeduren besteht darin, dass eine Funktion **immer** genau einen Wert zurückgibt, eine Prozedur **kann** einen oder mehrere Werte zurückgeben, muss es aber nicht.

1. Gemeinsamkeiten von Prozeduren und Funktionen

- Beide besitzen eine Kopfzeile beginnend mit dem Schlüsselwort **`function`** oder **`procedure`**.
- In beiden kann man vom Hauptprogramm unabhängige *lokale Variable* deklarieren. Diese sind dann nur innerhalb der Prozedur oder der Funktion bekannt! Im Gegensatz dazu gibt es auch die Möglichkeit im `implementation` – Teil des Hauptprogramms *globale Variable* zu deklarieren, die dann im gesamten Programm, einschließlich aller Unterprogramme, verwendet werden können.
- An beide können vom Hauptprogramm aus Werte (so genannte *Parameter*) übergeben werden. Die in der Funktion bzw. lokalen Prozedur verwendeten Parameter heißen *formale Parameter*. Sie werden beim Aufruf der Funktion bzw. Prozedur durch *aktuelle Parameter* ersetzt.
- In beiden ist der Quelltext durch die Schlüsselwörter **`begin`** und **`end`** geklammert.

2. Funktionen

Zunächst soll der allgemeine Aufbau einer Funktion verdeutlicht werden.



[1] Inhaltlich entnommen aus: Bähnisch, Uwe: *Praktische Informatik mit Delphi* Band 1, Cornelsen Verlag Berlin 2001

Eine Funktion besteht immer aus einem **Funktionskopf** und einem **Funktionskörper**.

Funktionskopf: Schlüsselwort - Funktionsname - formale Parameter - Rückgabewert
Funktionskörper: Berechnungsvorschrift, Wertrückgabe

Beispiel: Die Berechnung des Mittelwertes zweier Zahlen in Objekt Pascal (Delphi)

```
function mittelwert (x, y : real) : real;
begin
result := (x + y) / 2;
// mittelwert := (x + y ) / 2; ist ebenfalls moeglich
end;
```

Der Aufruf der Funktion erfolgt im Hauptprogramm, z.B. durch die folgende Anweisung:

```
z := mittelwert(zahl1, zahl2);
```

Bitte beachten Sie, dass die formalen Parameter `x` und `y` vollkommen verschieden sind von den aktuellen Parametern `zahl1` und `zahl2`. Es ist allerdings darauf zu achten, dass die formalen und die aktuellen Parameter in Anzahl, Reihenfolge und Datentyp übereinstimmen müssen. Natürlich ist die entsprechende Funktion im Quelltext des Programms **vor** dem Hauptprogramm zu definieren in dem sie aufgerufen werden soll.

Aufgabe:

1. Schreiben Sie ein Programm zur Berechnung des Mittelwertes zweier Zahlen. Die Berechnung des Mittelwertes soll dabei in einer eigenständigen Funktion erfolgen.

```
function mittelwert (x,y: real): real;
begin
  result := (x + y) / 2;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  zahl1, zahl2, ergebnis: real;
begin
  zahl1 := StrToFloat(Edit1.Text);
  zahl2 := StrToFloat(Edit2.Text);
  ergebnis := mittelwert(zahl1,zahl2);
  Edit3.Text := FloatToStrF(ergebnis,ffFixed,6,2);
end;
```

2. Schreiben Sie eine Funktion zur Berechnung des Maximums zweier übergebener Zahlen!

```
function maximum (x,y: real): real;
begin
  if x > y then result := x else result := y;
end;

procedure TForm1.Button1Click(Sender: TObject);
var
  zahl1, zahl2, ergebnis: real;
begin
  zahl1 := StrToFloat(Edit1.Text);
  zahl2 := StrToFloat(Edit2.Text);
  ergebnis := maximum(zahl1,zahl2);
  Edit3.Text := FloatToStrF(ergebnis,ffFixed,6,2);
end;
```

Funktion zur Berechnung des Maximums von vier übergebenen Zahlen

```
function maximum4(a,b,c,d: integer): integer;
var
  maximum: integer;
begin
  if a > b then maximum := a else maximum := b;
  if maximum < c then maximum := c;
  if maximum < d then maximum := d;
  result := maximum;
end;

procedure TForm1.Button4Click(Sender: TObject);
var
  zahl1, zahl2, zahl3, zahl4, max: integer;
begin
  zahl1 := StrToInt(Edit4.Text);
  zahl2 := StrToInt(Edit5.Text);
  zahl3 := StrToInt(Edit6.Text);
  zahl4 := StrToInt(Edit7.Text);
  max := maximum4(zahl1,zahl2,zahl3,zahl4);
  Edit8.Text := IntToStr(max);
end;
```