Prozeduren und Funktionen: ein kleines Übungsprojekt

Prolog

Prozeduren und Funktionen haben wir schon oft in Delphi genutzt, z. B. beim Erzeugen von Zufallszahlen mit den Anweisungen `Randomize;` und `Zahl := Random(42);` . Dabei ist **„Randomize“ eine Prozedur, „Random“ eine Funktion**. Der Unterschied besteht darin, dass eine Funktion immer einen Funktionswert zurückgibt (im Beispiel eine Zahl von 0 bis 41). Eine Prozedur besitzt hingegen keinen Rückgabewert.

Von Delphi erstellte Prozeduren begegnen uns bspw. immer dann, wenn wir ein OnClick-Ereignis definieren. So erstellt Delphi beim Doppelklick auf den Button „BtnTueWas“ automatisch folgenden Prozedurrumpf im Quelltext-Editor:

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

end;
```

Man muss sich nicht auf die von Delphi vorgegebenen Funktionen und Prozeduren beschränken, man kann auch eigene erstellen und nutzen.

Eigene Prozeduren erstellen und nutzen

In der Delphi-Unit zu diesem Übungsprojekt finden Sie ein Beispiel für eine selbst erstellte Prozedur:

```
procedure Zufallseingabe;
begin
  Form1.SpnEdtX.Value := Random(100);
  Form1.SpnEdtY.Value := Random(100);
end;
```

Die Prozedur weist den SpinEdits „SpnEdtX“ und „SpnEdtY“ Zufallswerte von 0 bis 99 zu.

Die Definition einer Prozedur beginnt mit dem Schlüsselwort `procedure` , gefolgt vom Namen der Prozedur. Nach dem Prozedurnamen kann eine Parameterliste stehen (s. u.). Die zur Prozedur gehörenden Anweisungen werden zwischen `begin` und `end` eingefasst.

Aufgabe 1

Öffnen Sie das Projekt „FundPUebungsprojekt“ in Delphi.

a) Der Button „BtnZufall“ (mit der Beschriftung „X und Y zufällig wählen“) soll den SpinEdits „SpnEdtX“ und „SpnEdtY“ zufällige Werte von 0 bis 99 zuordnen. Machen Sie

einen Doppelklick auf den Button und schreiben Sie `Zufallseingabe;` zwischen `begin` und `end` . Hierdurch wird die oben schon definierte Prozedur „Zufallseingabe“ aufgerufen und ihr Code ausgeführt. Testen Sie das Programm.

b) Betrachten Sie die Definition der Prozedur „Zufallseingabe“. Sie werden feststellen, dass die SpinEdits mit dem Präfix `Form1.` angesprochen werden. **Dies ist nötig, da in selbst erstellten Prozeduren und Funktionen nicht direkt auf Komponenten des Formulars (GUI) zugegriffen werden kann.** Testen Sie, was passiert, wenn man das Präfix `Form1.` vor `SpnEdt` entfernt.

c) Betrachten Sie den Quellcode für den Button „BtnReset“ („alle Ergebnisse löschen“). Die vier Anweisungen sollen in eine eigene Prozedur ausgelagert werden. Definieren Sie dazu eine Prozedur (unterhalb der Prozedur „Zufallseingabe“) mit dem Namen „EditInhalteLoeschen“, in die Sie die vier Anweisungen „verschieben“. Ersetzen Sie dann die Anweisungen in der Prozedur „BtnResetClick“ durch einen Aufruf der Prozedur „EditInhalteLoeschen“. Testen Sie das Programm.

d) Der Button „BtnZufall“ („X und Y zufällig wählen“) soll nach dem Erzeugen der Zufallseingaben auch alle Inhalte der Ergebnis-Edits löschen. Nutzen Sie hierzu wieder einen Aufruf der Prozedur „EditInhalteLoeschen“ (vgl. vorige Aufgabe).

Diese Aufgabe verdeutlicht einen wichtigen Vorteil selbst erstellter Prozeduren und Funktionen: **Quellcode kann für verschiedene Situationen wiederverwendet werden, ohne ihn redundant [≈ unnötig mehrfach] niederschrieben zu müssen**.

Intermezzo

Kommen wir nun zur oben angesprochenen Parameterliste. Diese erlaubt einen flexibleren Einsatz von Prozeduren (und Funktionen), da bestimmte Werte oder Variablen übergeben werden können. Zum Beispiel kann man die Prozedur

```
procedure SagPiep(Anzahl: Integer);
var
  i: Integer;
begin
  for i := 1 to Anzahl do
    ShowMessage('Piep!');
end;
```

mit der Anweisung `SagPiep(42);` aufrufen, um 42 mal „Piep“ auszugeben. Durch die Parameterliste (hier: `Anzahl: Integer`) wird die in der Prozedur verwendete Variable „Anzahl“ deklariert. Durch den Aufruf `SagPiep(42)` wird der Variablen „Anzahl“ der Wert 42 zugewiesen – man sagt auch, der Prozedur „SagPiep“ wird der Wert 42 übergeben.

Aufgabe 2

Beim Drücken des Buttons „BtnZaehle“ („zähle von X bis Y“) soll mit Hilfe von „ShowMessage“ von X bis Y gezählt werden. Schreiben Sie hierzu eine Prozedur „ZaehleVonBis“ mit

Parameterliste, und rufen Sie sie mit entsprechenden Werten in der Prozedur „BtnZaehle-Click“ auf.

Eigene Funktionen erstellen und nutzen

Funktionen unterscheiden sich von Prozeduren dadurch, dass sie einen Rückgabewert liefern. Unterhalb der Prozedur „ZaehleVonBis“ befindet sich folgende Funktion:

```
function Addiere(Summand1,Summand2: Integer): Integer;
begin
  Result := Summand1 + Summand2;
end;
```

Funktionen werden mit dem Schlüsselwort `function` definiert. Wie bei Prozeduren folgt der Name der Funktion und optional eine Parameterliste. Hinter der Parameterliste wird der Typ des Rückgabewertes der Funktion definiert (hier: Integer, aber es sind auch andere Typen wie String, Boolean etc. möglich). Der Rückgabewert wird in der Variablen „Result“ gespeichert. Diese Variable muss nicht deklariert werden, sondern steht in jeder Funktion automatisch zur Verfügung. Der Wert, der am Ende der Funktion in „Result“ steht, wird von der Funktion zurückgeliefert. Da Funktionen also Werte zurückgeben, kann ein Funktionsaufruf nicht für sich alleine stehen – der zurückgelieferte Wert muss (z. B. von einer Variablen) „in Empfang genommen“ werden.

Beispiel: Die Anweisung `Ergebnis := Addiere(4,5);` ruft die Funktion „Addiere“ auf, die am Ende den Wert 9 an die Variable „Ergebnis“ zurückliefert (und dieser zuweist).

Aufgabe 3

a) In der Prozedur „BtnPlusClick“ zum Button „BtnPlus“ („X + Y“) stimmt etwas nicht (testen Sie z. B. die Berechnung für x = 3 mit y = 5). Korrigieren Sie den Fehler.

b) Implementieren Sie eine Funktion „Multipliziere“, und nutzen Sie sie in der Prozedur „BtnMalClick“ („X * Y“).

c) Implementieren Sie eine Funktion „Potenziere“ („X ^ Y“, sprich: „X hoch Y“) oder „Fakultaet“ („X!“), und nutzen Sie sie in der Prozedur des entsprechenden Buttons.

Tipps:
- Nutzen Sie für die Berechnung der Potenz und der Fakultät jeweils eine For-Schleife.
- Beachten Sie bei der Potenz den Sonderfall y = 0, nämlich $x^0 = 1$.
- Die Fakultät von x ist definiert als ist das Produkt von 1 • 2 • 3 • … • x. Es gilt der Sonderfall 0! = 1 (Näheres siehe z. B. Wikipedia).

Aufgabe 4

a) Worin liegt der Unterschied zwischen Prozeduren und Funktionen?
b) Welchen Nutzen bieten Prozeduren und Funktionen?