Rekursion

Mit Hilfe der Babuschka

Schon die Russen beschäftigten sich mit der Verschachtelung von Dingen. Eine Babuschka-Figur lässt sich öffnen und innen befindet sich dann eine nächste, kleinere Babuschka-Figur und irgendwann gibt es keine weitere, kleine und man beginnt sie wieder zusammenzusetzen.

Ineinander verschachtelte Babuschka-Figuren

Einzelne Funktionen in der Mathematik können ebenso verschachtelt dargestellt werden. Zum Beispiel wird die Fakultät mit folgenden Regeln rekursiv definiert:

a) 0! = 1

b) 1! = 1

c) n! = n * (n – 1)! , wenn n > 1

Man beginnt eine Fakultät *n!* mit *n > 1* wie die Babuschka-Figur zu öffnen und findet darin eine weitere Fakultät. Irgendwann ist man bei der letzten, kleinsten Figur *1!* angelangt und man kann beginnen, das Resultat rückwärts wieder zusammenzusetzen.

Mit Hilfe der Rekursion können wir also Probleme verschachtelt definieren. Die beiden ersten Regeln a) und b) definieren die "untersten" Resultate (Rekursionsschluss), wobei die dritte Regel c) einen beliebigen Wert n > 1 nimmt und das Resultat über das Delegieren an einen weiteren Funktionsaufruf berechnet (Rekursionsschritt). Dieser Aufruf wird Rekursionaufruf genannt. Die oben aufgeführte Fakultät-Funktion ist rekursiv definiert und kann in Eiffel folgendermassen implementiert werden:

```
feature
        rek_fak( n : INTEGER ) : INTEGER is
                -- rekursiv Fakultät berechnen
do
        if ( n < 2 ) then
                Result := 1
        else
                Result := n * rek_fak( n – 1)
        end
end
```

Sämtliche Rekursionen können zudem immer auch iterativ (ohne Rekursion, nur mit Schleifen) berechnet werden (Die theoretische Informatik hat den Beweis dazu, auf welchen hier nicht näher eingegangen werden soll):

```
feature
        fak( n : INTEGER ) : INTEGER is
                -- iterativ Fakultät berechnen
local
        i : INTEGER
do
        from
                i := 1
        until
                i = n
```

```
        loop
                Result := Result * i
                i := i + 1
        end
end
```

Beim Schreiben von rekursiven Funktionen müssen die beiden folgenden Punkte erfüllt sein:

- es existiert für jeden Fall ein Rekursionsschluss
- für Werte, die nicht mit dem Rekursionschluss abgeschlossen werden, existiert ein Rekursionsschritt

In der Mathematik haben wir diese zwei Dinge auch bei der vollständigen Induktion: wir benötigen eine Verankerung (Rekursionschluss) und den Induktionsschritt (Rekursionsschritt).

Übungen zur Rekursion

Binärdarstellung

Wir möchten aus einer Zahl im Dezimalformat die Binärdarstellung berechnen und ausgeben. Dazu haben wir das folgende Feature vorgegeben:

```
feature
        dig( dez : INTEGER ) is
                -- Binärdarstellung anzeigen
        do
                ...
                io.put_integer( ... )
        end
```

- Überlege zuerst, wie die Binärdarstellung definiert wird.
- Versuche das Problem zusätzlich auch noch iterativ zu lösen

Tipp

Die Modulofunktion a mod b wird in Eiffel folgendermassen geschrieben:

```
a \\ b
```

Ganzzahlig dividieren kann man mit:

```
a // b
```

Quersumme

Berechne die Quersumme einer Zahl.

Tipp

Berechne den Wert der ersten Ziffer und rekursiv die des Rests.

Modulofunktion

Implementiere die Modulofunktion eigenhändig.

```
feature
        mod( a, b : INTEGER ) : INTEGER is
```

Gerade oder ungerade

Eruiere, ob eine gegebene Zahl gerade oder ungerade ist.

```
feature
        par( a : INTEGER ) : BOOLEAN is
```

Tipp

Vergleiche das Problem mit der Fakultätsaufgabe – beide Aufgaben sind äusserst ähnlich.

Die Türme von Hanoi

Die Aufgabe besteht darin, den Turm aus verschieden großen Scheiben, von der linken auf die rechte Unterlage zu verschieben. Es kann immer

nur eine Scheibe transportiert werden und zwar die oberste eines jeden Stapels. Ferner darf eine Scheibe nur abgelegt werden, wenn der Platz ganz frei ist, oder eine größere Scheibe dort zu oberst liegt.

Folgendes Feature soll implementiert werden:

```
feature
        hanoi( hoehe, stab1, stab2, stab3 : INTEGER )
do
        if ( hoehe = 1 ) then
                io.put_string( "bewege von " )
                io.put_integer( stab1 )
                io.put_string(" nach " )
                io.put_integer(stab3 )
                io.put_new_line
        else
                -- … einige rekursive Aufrufe …
        end
```

Das Programm wird gestartet mit:

```
        hanoi( anzahlscheiben, 1, 2, 3 )
```

Tipp

Die Argumente *stab1*, *stab2* und *stab3* werden jeweils beim rekursiven Aufruf so vertauscht, dass immer in der nächsten Rekursionstufe die richtige Vertauschung gemacht wird. Zudem sind mehrere rekursive Aufrufe nötig.

Referenzen

Jobware Babuschka Bild – http://www.jobware.de/view/gbkn74k16n.html

Aufgaben – http://pimpf.pi.informatik.tu-darmstadt.de/loop/aufgaben/turban/own/