Iteration und Rekursion

Unterscheidung:

Unter **Iteration** versteht man ein mehrmaliges Ausführen einer Aktion. Typischerweise zählt man dabei mit, zum wievielten Mal die Aktion jeweils erledigt wird (muss aber nicht sein).

Stichwort: „Schleife“; einige Schlüsselworte: „for“, „while“, „repeat“, „loop“

Pizzaessen :
    vom ersten bis zum letzten Bissen:
        schneide diesen Bissen ab und iss ihn auf
    // Ende der Schleife
    FERTIG
// Ende von Pizzaessen

Unter **Rekursion** versteht man ebenfalls eine wiederholte Ausführung, doch diesmal gibt es keine Zählschleife, sondern die Aktion erklärt sich „durch sich selbst“.

Stichwort: „Selbstbezug“, „Selbstaufruf“

Pizzaessen :
    wenn *Teller leer* : FERTIG
    sonst:
        schneide einen Bissen ab und iss ihn auf
        Pizzaessen
    // Ende von wenn
// Ende von Pizzaessen

Hier ruft sich die Funktion *Pizzaessen* ständig selbst auf. „Immer wieder *Pizzaessen*, bis der Teller leer ist“. Der Vorgang stoppt irgendwann, da bei jedem Ablauf das Pizzastück kleiner wird.

Ein weiteres Beispiel: **zählen von 1 bis 10:** $count(1, 10)$

Iteration:

$count(zahl, max)$:
    wiederhole solange $zahl \leq max$:
        Ausgabe: $zahl$
        $zahl := zahl + 1$
    // Ende der Schleife
// Ende von count

Rekursion:

$count(zahl, max)$:
    wenn $zahl > max$: FERTIG
    sonst:
        Ausgabe: $zahl$
        $count(zahl + 1, max)$
    // Ende von wenn
// Ende von count

Erkennbar wird hier der typische Aufbau einer rekursiven Funktion:

1. eine Abbruchbedingung, die das Ende der Selbstaufrufe bestimmt
2. ein Aufruf der Funktion selbst

Oftmals lassen sich Aufgaben leichter rekursiv formulieren, als iterativ. Eine Rekursion ist dann praktisch, wenn man ein-und-dieselbe Aktion immer wieder durchführt und dabei eine Aufgabe immer mehr erledigt, bis sie gelöst ist.

Einige Beispiele für rekursive Funktionen

Beispiel 1:

$$
\begin{array}{l}
a \in \mathbb{R}, n \in \mathbb{N} \\
p(a,n) \quad := \quad \left\{ \begin{array}{ll} 1 & \text{, falls } n = 0 \\ a \cdot p(a, n-1) & \text{, sonst} \end{array} \right.
\end{array}
$$

Was tut diese Funktion?

Vielleicht sieht man es hier schon:

Mit jedem Aufruf wird einmal mit a multipliziert. Jedesmal wird der zweite Parameter (n) dabei um 1 reduziert (das geschieht offensichtlich n mal) bis der zweite Parameter schließlich die 0 erreicht. Beim letzten Mal wird schließlich nur mit 1 multipliziert.

Ergebnis ist:

$$
p(a,n) \quad = \quad \underbrace{a \cdot a \cdot a \cdot \ \ldots \ \cdot a}_{n \text{ mal}} \cdot 1 \quad = \quad a^n
$$

Beispiel 2:

$$
\begin{array}{l}
a, b \in \mathbb{N} \\
d(a,b) \quad := \quad \left\{ \begin{array}{ll} 0 & \text{, falls } a < b \\ d(a-b, b) + 1 & \text{, sonst} \end{array} \right.
\end{array}
$$

Was tut diese Funktion?

Hier ist es nicht ganz so leicht zu sehen. Also setzen wir einmal für ein paar Beispiele in die Funktion Werte ein und vollziehen nach, was passiert:

$$
\begin{array}{rcl}
d(3,4) & = & 0 \\
\\
d(10,3) & = & d(7,3) + 1 \\
& = & \big(d(4,3) + 1\big) + 1 \\
& = & \big(\big(d(1,3) + 1\big) + 1\big) + 1 \\
& = & 0 + 1 + 1 + 1 \quad = \quad 3
\end{array}
$$

Hier tritt der erste Fall ein ($a < b$). Allgemein wird das Ergebnis immer 0 sein, wenn der erste Parameter kleiner als der zweite ist.

Wiederholtes Einsetzen in die Funktion ergibt eine Reihe von 1-en, die aufsummiert werden. Mit jedem Schritt wird b von a abgezogen.

Letzendlich wird auf diese Weise mitgezählt, wie oft b in a „hineinpasst“? — Dies entspricht der Definition der ganzzahligen Division.

Beispiel 3:

$$
\begin{array}{l}
k, n \in \mathbb{N} \\
b(n,k) \quad := \quad \left\{ \begin{array}{ll} 1 & \text{, falls } k = 0 \text{ oder } k = n \\ b(n-1, k-1) + b(n-1, k) & \text{, falls } 1 < k < n \end{array} \right.
\end{array}
$$

Was tut diese Funktion?

Hier ist es ebenfalls nicht leicht zu sehen. Von der Struktur her lässt sich jedoch zumindest folgendes sagen:

- Die Rekursion ist baumartig, d.h. um den Wert der Funktion zu bestimmen, müssen mehrere (hier jeweils 2) „Vorgänger“ bestimmt werden — bei jedem rekursiven Aufruf.
- Bei gegebenen n (1. Parameter) bewegt sich k (2. Parameter) immer in den Grenzen $0 \leq k \leq n$. Dies weist auf eine Dreiecksstruktur hin (k wird nie größer als n).

In einer anderen Schreibweise erscheint die Funktion vielleicht etwas vertrauter:

$$
\tbinom{n}{k} := \tbinom{n-1}{k-1} + \tbinom{n-1}{k}, \quad \tbinom{n}{0} := 1, \quad \tbinom{n}{n} := 1
$$

Es handelt sich um die *Binomialkoeffizienten*. (Stichwort: Pascal’sches Dreieck)

Beweis durch vollständige Induktion

Beispiel 1: Binomialkoeffizienten

Für die Binomialkoeffizienten gibt es auch noch eine andere Formel zur Berechnung:

$$\binom{n}{k} \;=\; \frac{n!}{(n-k)!\cdot k!}$$

Um zu beweisen, dass die beiden Berechnungsmethoden übereinstimmen, kann man die Methode der vollständigen Induktion anwenden. Der Beweis durch vollständige Induktion bietet sich gerade für rekursive Formeln an, da die Struktur der Beweismethode und der Rekursion ähnlich sind.

Ind. Beh.: $b(n,k) \equiv \binom{n}{k} \stackrel{!}{=} \binom{n-1}{k-1} + \binom{n-1}{k} \equiv b(n-1,k-1)$, und es gilt: $\binom{n}{k} = \frac{n!}{(n-k)!\cdot k!}$
sowie $b(0,0) = 1, \quad b(n,0) = 1, \quad b(n,n) = 1$

Ind. Anf.: $b(0,0) = \frac{0!}{(0-0)!\cdot 0!} = \frac{1}{1} = 1$
$b(n,0) = \frac{n!}{(n-0)!\cdot 0!} = \frac{n!}{n!\cdot 1} = 1\,, \qquad b(n,n) = \frac{n!}{(n-n)!\cdot n!} = \frac{n!}{n!}\cdot\frac{1}{0!} = 1$

Ind. Schritt: $(n \to n+1)$

Annahme: Die Behauptung sei erfüllt für n.
zu zeigen: Die Behauptung gilt dann auch für $n+1$

Umgesetzt auf den konkreten Fall ist zu zeigen:
($n+1$ dort einsetzen, wo in der Behauptung n stand)

$$\begin{aligned} b(n+1,k) &\stackrel{!}{=} b((n+1)-1,k-1) + b((n+1)-1,k) \\ &= b(n,k-1) + b(n,k) \end{aligned}$$

Beweis:

$$\begin{aligned} b(n,k-1) + b(n,k) &= \frac{n!}{(n-(k-1))!\cdot(k-1)!} + \frac{n!}{(n-k)!\cdot k!} \\ &= \frac{n!}{(n-k)!\cdot(n-k+1)\;\cdot\;(k-1)!} + \frac{n!}{(n-k)!\cdot(k-1)!\cdot k} \\ &= \frac{n!}{(n-k)!\cdot(n-k+1)\cdot(k-1)!\quad\cdot k} + \frac{n!\qquad\cdot(n-k+1)}{(n-k)!\cdot(k-1)!\cdot k\quad\cdot(n-k+1)} \\ &= \frac{n!\cdot k \;+\; n!\cdot(n-k+1)}{(n-k)!\cdot(k-1)!\cdot(n-k+1)\cdot k} = \frac{n!\cdot(k+(n-k+1))}{(n-k)!\cdot(n-k+1)\;\cdot\;(k-1)!\cdot k} \\ &= \frac{n!\cdot(n+1)}{(n-k+1)!\cdot(k)!} = \frac{(n+1)!}{((n+1)-k)!\cdot(k)!} = b(n+1,k) \end{aligned}$$

Beispiel 2: Knoten und Kanten eines Baumes

Zu Zeigen ist, dass ein Baum mit n Knoten immer $n-1$ Kanten hat.
Def eines (ungerichteten) Baumes: Ein azyklischer, zusammenhängender Graph.

Induktionsannahme: Ein Baum mit n Knoten hat $n-1$ Kanten. Induktionsanfang:

$n=1$: Ein Baum mit 1 Knoten hat 0 Kanten.
$n=2$: Ein Baum mit 2 Knoten hat 1 Kante.

Induktionsschritt: $(n \to n+1)$

Wir haben einen Baum mit n Knoten und $n-1$ Kanten. Wir fügen einen Knoten zu diesem Baum hinzu, also haben wir schon $n+1$ Knoten. Die Frage ist nun, wieviele Kanten können denn nun hinzugefügt werden. Machen wir doch eine Fallunterscheidung:

0 Kanten: Der Knoten wird gar nicht mit dem Baum verbunden. Das ist nicht zulässig, denn ein Baum ist definiert als ein zusammenhängender Graph. → nicht möglich.

$\geq$ 2 Kanten: Knoten $n+1$ wird über eine Kante mit einem beliebigen Knoten i des Baumes verbunden und dann noch mit einem Knoten $j \neq i$ (weitere Kanten spielen keine Rolle). Laut Definition vom Baum ist der Graph zusammenhängend ⇒ es ex. ein Weg von i nach j. Zwei Kanten wurden hinzugefügt: $(n+1,j)$ und $(n+1,i)$. Folglich wurde ein Kreis gebildet $n+1 \to j \to i \to n+1$. Laut Definition vom Baum muss der Graph azyklisch sein ⇒ es ist nicht möglich mehr als eine Kante hinzuzufügen ⇒ die Anzahl der Kanten ist um eins größer geworden, beträgt also n. **qed**.