griffe auf die Speicherplätze eines Arrays notwendig sind.

Im Allgemeinen laufen iterative Methoden bedeutend rascher als rekursive Methoden.

6. Die Ackermann-Funktion

Der Mathematiker **Wilhelm Ackermann** definierte eine Folge, deren Glieder selbst Funktionen aus den vorhergehenden Gliedern bilden (vgl. [7]). Die von der ungarischen Mathematikerin **Rósza Péter** vereinfachte Version verwenden wir für das nächste Beispiel. Dabei ist definiert:

```
a(0,m) = m + 1
a(n,0) = a(n-1,1)
a(n,m) = a(n-1, a(n, m-1))
```

```
public class Ackermann {
  static long ackermann(long n, long m) {
        if (n==0) return m+1;
        else if (m==0) return ackermann(n-1,1);
        else return ackermann(n-1,
ackermann(n,m-1));
  }
  public static void main (String [] args) {
    int x = Integer.parseInt(args[0]);
    int y = Integer.parseInt(args[1]);
    System.out.println("a(" + x + "," + y + ")
= " + ackermann(x,y));
  }
}
```

Vorsicht: Beim Berechnen der Ackermann-Funktion stellt sich heraus, dass die Berechnungen sehr aufwändig werden. Bereits bei kleinen Argumenten führen die Rekursionen zu einem so genannten „Stacküberlauf" - die Rekursionstiefe wurde zu groß.

```
nus@ice:~/java_bsp> java Ackermann 3 10
a(3,10) = 8189
nus@ice:~/java_bsp> java Ackermann 4 1
Exception in thread "main"
java.lang.StackOverflowError
```

7. Legendär: Die Türme von Hanoi

Die (legendären) Türme von Hanoi bestehen aus durchbohrten Scheiben, die der Größe nach auf Säulen geschlichtet sind. Auf einer Säule befinden sich n Scheiben - sie sollen Stück für Stück auf eine andere Säule umgeschlichtet werden, wobei eine Scheibe immer nur auf einer größeren Scheibe zu liegen kommen darf. Damit dies möglich ist, können Scheiben zwischenzeitlich auf einer dritten Säule gestapelt werden - aber auch auf dieser Säule darf stets eine Scheibe nur auf einer größeren zu liegen kommen.

Im folgenden Beispiel wird das Umschlichten der Scheiben rekursiv berechnet und Schritt für Schritt ausgegeben:

```
public class Hanoi {
  static long schritte;
  static void lege(int n, char von, char nach,
char zwischen) {
    if (n > 0) {
      lege (n-1, von, zwischen, nach);
      System.out.println(n + ". Scheibe von " +
von + " nach " + nach);
      lege (n-1, zwischen, nach, von);
      schritte ++;
    }
  }
  public static void main (String [] args) {
    int maxzahl = Integer.parseInt(args[0]);
    lege (maxzahl, 'a', 'c', 'b');

System.out.println("--------------------------
-------------");
    System.out.println(schritte + " Schritte");
  }
}
```

Damit erhalten wir für 4 Scheiben beispielsweise folgende „Spielzüge":

```
nus@ice:~/java_bsp> java Hanoi 4
1. Scheibe von a nach b
2. Scheibe von a nach c
1. Scheibe von b nach c
3. Scheibe von a nach b
1. Scheibe von c nach a
2. Scheibe von c nach b
1. Scheibe von a nach b
4. Scheibe von a nach c
1. Scheibe von b nach c
2. Scheibe von b nach a
1. Scheibe von c nach a
3. Scheibe von b nach c
1. Scheibe von a nach b
2. Scheibe von a nach c
1. Scheibe von b nach c
---------------------------------------
15 Schritte
```

Mit 4 Münzen lässt sich dies sehr rasch „nachspielen". Wie viele Schritte sind für 5, 6, ... Scheiben erforderlich? Wie lange dauert das Umschlichten, wenn man für das Umlegen einer Scheibe 2 Sekunden benötigt?

8. Ausblick, Übungen

Rekursive Algorithmen werden in allen Programmierumgebungen verwendet.

- Formulieren Sie eine rekursive Funktion zur Berechnung von Potenzen `f(x) = xk`!
- Berechnen Sie die Summe `1 + x-1 + x-2 + ... + x-n` rekursiv und iterativ!
- Formulieren Sie ein Programm zur Berechnung der Fibonaccizahlen, das ohne rekursive Methodenaufrufe auskommt!
- Untersuchen Sie das Verhalten der Funktionswerte, wenn Sie die Fibonaccifolge in einer Restklasse (z.B. `mod 7`) berechnen!
- Der größte gemeinsame Teiler kann mit der folgenden rekursiv aufgerufenen Methode berechnet werden:

```
static int ggT(int a, int b) {
  if (a == b) return a;
  else if (a < b) return ggT(a,b-a);
  else return ggT(a-b,b);
}
```

Formulieren Sie eine entsprechende JAVA-Klasse!

- Mit Rekursionen lassen sich ansprechende Grafiken erzeugen. Recherchieren Sie dazu Beispiele aus der fraktalen Geometrie (zB [1])!

8. Literatur, Weblinks

[1] PCNEWS-76, Feb. 2002, S. 49, „JAVA - Fraktale"

[2] PCNEWS-98, , S. 25, „JAVA – Verschiedene Ausgaben auf der Konsole"

[3] **Guido Krüger**, „Handbuch der JAVA-Programmierung", Addison & Wesley, ISBN 3-8273-2201-4

[4] *http://www.javabuch.de* („Handbuch der JAVA-Programmierung", freier Download für schulische Zwecke)

[5] **Herbert Schildt**, „JAVA – Grundlagen der Programmierung", mitp, IBSN 3-8266-1524-7

[6] **Udo Müller**, „JAVA – das Lehrbuch", mitp, ISBN 3-8266-1333-3

[7] **Christian Ullenboom**, „Java ist auch eine Insel", Galileo Computing, ISBN 3-89842-365-4

[8] *http://www.gymmelk.ac.at/nus/informatik/wpf/JAVA* (Unterrichtsbeispiele zum Programmieren mit JAVA, Quelltexte zum Downloaden)

CC | Skripten

Martin Weissenböck

ADIM, Arbeitsgemeinschaft für Didaktik, Informatik und Mikroelektronik
1190 Wien, Gatterburggasse 7
Tel.: 01-369 88 58-88
FAX: 01-890 01 21-77

EDV-Skripten

Nr	Titel
38	Turbo Pascal (Borland)
39	RUN/C Classic
40	Turbo-C (Borland)
41-3	Turbo/Power-Basic
43-2	DOS
43-3	DOS und Windows
47	Turbo-Pascal (Borland)
49	Quick-Basic (Microsoft)
50	C++ (Borland)
53-3	AutoCAD I (2D-Grafik)
53-5	AutoCAD I (2D-Grafik)
54	AutoCAD II (AutoLisp+Tuning)
55	AutoCAD III (3D-Grafik)
56	Grundlagen der Informatik
61	Visual Basic (Microsoft)
63	Windows und Office
81	Linux
191,192	Angewandte Informatik I + II
201,202	Word I+II
203	Excel
205,206	Access I+II
221	HTML
222	HTML und CSS
223	JavaScript,
227	VB.NET
231,232	Photoshop I+II
237	Dreamweaver

CDs

Nr	Titel
110	Best Of VoIP (CD)
111	All About VoIP (DVD)

Bestellhinweise

http://www.adim.at/

http://www.adim.at/