EINI LW

Einführung in die Informatik für Naturwissenschaftler und Ingenieure

Vorlesung 2 SWS WS 11/12

Dr. Lars Hildebrand
Fakultät für Informatik – Technische Universität Dortmund
lars.hildebrand@udo.edu
http://ls1-www.cs.uni-dortmund.de

technische universität dortmund

Dr. Lars Hildebrand

- **Kapitel 4**
  Grundlagen imperativer und objektorientierter Programmierung:
  - **Funktionen**
  - **Prozeduren**
  - **Methoden**
  - **Rekursion**

- **Unterlagen**
  - Gumm/Sommer, Kapitel 2.7 & 2.8
  - Echtle/Goedicke, Einführung in die Programmierung mit Java, dpunkt Verlag, Kapitel 4
  - Doberkat/Dißmann, Einführung in die objektorientierte Programmierung mit Java, Oldenbourg, Kapitel 3.4 & 4.1

Zwischenstand

Dr. Lars Hildebrand

- Variablen
  - Bezeichner, Datentyp, Speicherort, Wert
- Zuweisungen
  - Zuweisungen müssen typverträglich sein
- (Einfache) Datentypen und Operationen
  - `integer (byte, short, long; float, double)`
  - `boolean`
  - `char`
  - `String`
  - Typkompatibilität
- Kontrollstrukturen
  - Sequentielle Komposition, Sequenz
  - Alternative, Fallunterscheidung
  - Schleife, Wiederholung, Iteration
- Verfeinerung
  - **Unterprogramme, Prozeduren, Funktionen**
  - Blockstrukturierung
- **Rekursion**

technische universität dortmund

Dr. Lars Hildebrand

Was ist also Rekursion?

- Eine Funktion f ist **rekursiv**, wenn
  - der Funktionsrumpf einen Aufruf der Funktion f selbst enthält oder einer Funktion g, die wiederum f aufruft.
  - Eine Terminierungsbedingung existiert.
  - Jede Eingabe nach endlich vielen Schritten terminiert.

technische universität dortmund

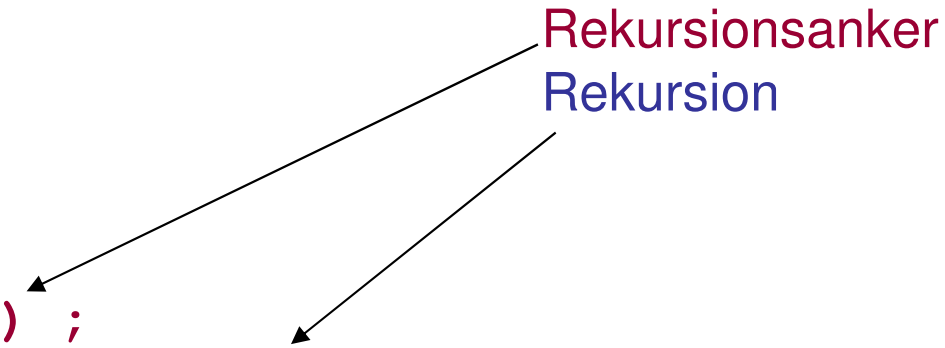
Dr. Lars Hildebrand

Beispiel: Fakultätsfunktion

- mathematische Definition

$$n! = \begin{cases} 1, & \text{falls } n = 0 \\ n \cdot (n-1)!, & \text{falls } n > 0 \end{cases}$$

- rekursive Funktion




```
int fakultaet(int n)
{
      if (n = = 0)    return(1) ;
      if (n > 0)      return( n * fakultaet(n-1) ) ;
}
```

wechselseitige Rekursion

- Eine **wechselseitige Rekursion** liegt vor, wenn eine Funktion f eine Funktion g aufruft, die ihrerseits wiederum f aufruft.

- Wechselseitige Rekursion ist natürlich auch mit mehr als 2 Funktionen möglich.

Direkte Rekursion

- Eine **direkte Rekursion** liegt vor, wenn eine Funktion f sich direkt selbst aufruft

Definition

- Eine rekursive Methode heißt **primitiv rekursiv**, wenn sie folgendes Schema hat, das auf den natürlichen Zahlen für n beruht.
- Schema

$$f(\mathrm{n}) = \begin{cases} s_0, & \text{falls } \mathrm{n} = 0 \\ h(\mathrm{n}, f(\, pred(\mathrm{n})\,)), & \text{sonst} \end{cases}$$

- basiert auf den Peano-Axiomen der natürlichen Zahlen
  - *h* ist eine Funktion,
  - *pred*(n) ist die Vorgängerfunktion.

technische universität dortmund

Dr. Lars Hildebrand

Beispiele zu primitiv rekursiven Funktionen

- Die Fakultätsfunktion ist **primitiv rekursiv**

$$n! = \begin{cases} 1, & \text{falls } n = 0 \\ n \cdot (n-1)!, & \text{falls } n > 0 \end{cases}$$

$$f(n) = \begin{cases} s_0, & \text{falls } n = 0 \\ h(n, f(\mathit{pred}(n))), & \text{sonst} \end{cases}$$

- $f(n) = n!$
- $s_0 = 1$
- $h(x, y) = *(x, y)$
  - übliche Schreibweise: $h(x, y) = x * y$
- $\mathit{pred}(n) = n - 1$

Beispiele zu primitiv rekursiven Funktionen

- Die Potenzfunktion ist **primitiv rekursiv**

$$x^n = \begin{cases} 1, & \text{falls } n = 0 \\ x \cdot x^{n-1}, & \text{falls } n > 0 \end{cases}$$

$$f(x,n) = \begin{cases} s_0, & \text{falls } n = 0 \\ h(x, f(\mathit{pred}(n))), & \text{sonst} \end{cases}$$

- $f(x, n) = x^n$
- $s_0 = 1$
- $h(x, y) = *(x, y)$
  - übliche Schreibweise: $h(x, y) = x * y$
- $\mathit{pred}(n) = n - 1$

technische universität dortmund

Dr. Lars Hildebrand

Definition

- Eine rekursive Methode heißt **endrekursiv** (tail recursive), wenn sie folgendes Schema hat.

- Schema

$$f(\mathrm{x}) = \begin{cases} g(\mathrm{x}), & \text{falls } P(\mathrm{x}) \\ f(\,r(\mathrm{x})\,), & \text{sonst} \end{cases}$$

- Endrekursiv, weil f() die letzte Aktion im rekursiven Zweig ist.

technische universität dortmund

Dr. Lars Hildebrand

Endrekursive Rekursion

- Beobachtung:
  - $f(x) = f(r(x))$
    $= f(r(r(x))) = f(r^2(x))$
    $= \ldots$
    $= f(r^k(x)) = g(r^k(x))$
- wobei k die kleinste natürliche Zahl ist, für die $P(r^k(x))$=true

- Daher einfaches iteratives Programm

```
f_iterativ(double x)
{
        while (!P(x)) x = r(x);
        return(g(x));
 }
```

technische universität dortmund

Dr. Lars Hildebrand

Beispiel: Fakultätsfunktion

- primitiv rekursive Fakultät

```
int fakultaet(int n) {
      if (n = = 0)    return(1) ;
      if (n > 0)      return( n * fakultaet(n-1) ) ;
}
```

- endrekursive Fakultät

```
int er_fakultaet(int n, int zwischenwert) {
      if (n <= 1)    return (zwischenwert) ;
      if (n > 1)     return (er_fakultaet(n-1, n*zwischenWert);
}
int fakultaet(int n) {
       return (er_fakultaet(n, 1));
```

technische universität dortmund

Dr. Lars Hildebrand

Beispiel: Fakultätsfunktion

- endrekursive Fakultät

```
int er_fakultaet(int n, int zwischenwert) {
      if (n <= 1)    return (zwischenwert) ;
      if (n > 1)     return (er_fakultaet(n-1, n*zwischenWert);
}
int fakultaet(int n) {
      return (er_fakultaet(n, 1));
```

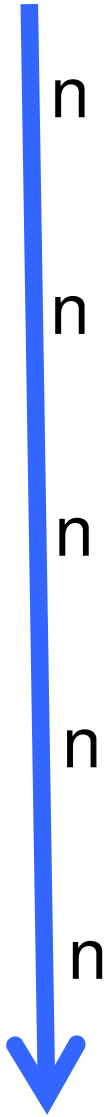
```
fakultaet(4) = er_fakultaet(4,1)
   er_fakultaet(4,1) = er_fakultaet(3, 1*4)
      er_fakultaet(3,4) = er_fakultaet(2, 4*3)
         er_fakultaet(2,12) = er_fakultaet(1, 12*2)=24
```

Zum Vergleich: primitiv-rek. Variante

Dr. Lars Hildebrand

4! =

```
int fakultaet(int n) {
     if (n = = 0)    return(1) ;
     if (n > 0)      return( n * fakultaet(n-1) ) ;
}
```



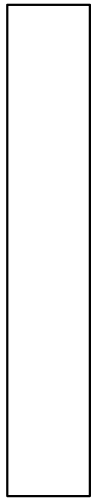
n = 4 n != 0 **return(n * fakultaet(n-1))**

n = 3 n != 0 **return(n * fakultaet(n-1))**

n = 2 n != 0 **return(n * fakultaet(n-1))**

n = 1 n != 0 **return(n * fakultaet(n-1))**

n = 0 n == 0 **return(1)**



primitiv-rek. Variante

Dr. Lars Hildebrand

4! =

```
int fakultaet(int n) {
     if (n = = 0)    return(1) ;
     if (n > 0)      return( n * fakultaet(n-1) ) ;
}
```

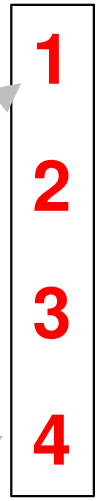
n = 4 n != 0 **return(n * fakultaet(n-1))**

n = 3 n != 0 **return(n * fakultaet(n-1))**

n = 2 n != 0 **return(n * fakultaet(n-1))**

n = 1 n != 0 **return(n * fakultaet(n-1))**

n = 0 n == 0 **return(1)**



technische universität dortmund

Dr. Lars Hildebrand

Beispiel: Fakultätsfunktion

- endrekursive Fakultät

```
fakultaet(4) = er_fakultaet(4,1)
   er_fakultaet(4,1) = er_fakultaet(3, 1*4)
      er_fakultaet(3,4) = er_fakultaet(2, 4*3)
         er_fakultaet(2,12) = er_fakultaet(1, 12*2)=24
```

- primitiv-rekursive Fakultät

```
fakultaet(4) = 4*fakultaet(3)          ->      fakultaet(4)=4*6=24
  fakultaet(3) = 3*fakultaet(2)        ->      fakultaet(3)=3*2=6
    fakultaet(2) = 2*fakultaet(1)      ->    fakultaet(2)=2*1=2
      fakultaet(1) = 1*fakultaet(0)    -> fakultaet(1)=1*1=1
                     fakultaet(0) = 1
```

Endrekursive Rekursion

- Umformung in iteratives Programm

```
f_iterativ(double x)
{
       while (!P(x)) x = r(x);
       return(g(x));
 }
```

```
public static int fakultaet_iterativ(int n) {
    int fakultaet = 1, zaehler = 1;

    while (zaehler <= n) {
      fakultaet = fakultaet * zaehler;
      zaehler++;
    }
    return(fakultaet);
  }
```

Vollständiges Beispiel

Dr. Lars Hildebrand

```
import java.util.Scanner;

public class Fak_iter
{
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    int eingabe = scanner.nextInt();
    int fak = fakultaet_iterativ(eingabe);
    System.out.println("Die Fak. von " + eingabe + " ist: " + fak);
  }

  public static int fakultaet_iterativ(int n) {
    int fakultaet = 1, int zaehler = 1;

    while (zaehler <= n) {
      fakultaet = fakultaet * zaehler;
      zaehler++;
    }

    return(fakultaet);
  }
}
```

technische universität dortmund

Dr. Lars Hildebrand

Definition

- Eine rekursive Methode heißt **linear**, wenn in den einzelnen Zweigen der bedingten Anweisung, die die Rekursion steuert, jeweils höchstens ein Aufruf der Methode vorkommt.
- Schema
-

$$f(\mathrm{x}) = \begin{cases} g(\mathrm{x}), & \text{falls } P(\mathrm{x}) \\ h(\mathrm{x}, f(\, r(\mathrm{x})\,)), & \text{sonst} \end{cases}$$

- Verallgemeinerung des
  - primitiv rekursiven Schemas: r = pred(x)
  - und der Endrekursion: h(x,y)=y

technische universität dortmund

Dr. Lars Hildebrand

Lineare Rekursion

- Beobachtung:
  - $f(x) = h(x,f(r(x)))$
    $= h(x, h(r(x), f(r(r(x))))) = h(x, h(r(x), f(r^2(x))))$
    $= ...$
    $= h(x, h(r(x), h(r^2(x), h(..., h(r^{k-1}(x),g(r^k(x)))...),$

- wobei k die kleinste natürliche Zahl ist,
  für die $P(r^k(x))$=true ist

- letzter Funktionsaufruf enthält k+1 Argumente

Transformation linear rekursiver Funktionen in iterative nicht-rekursive Funktionen

iteratives Programmfragment zur Implementierung einer linear rekursiven Funktion f(x)

```
stack s ;
while ( !P(x) )
{
    s.push(x)  ;
    x = r(x) ;
}
f = g(x) ;
while (!s.empty())
{
  x = s.pop() ;
  f = h(x,f) ;
}
```

stack s : Stapel mit Operationen

push(x): legt x oben auf den Stapel

pop(): liefert oberstes Element und entfernt es vom Stack

empty(): liefert true gdw Stapel ist leer

Stack muss implementiert werden.

Fibonacci-Zahlen

- Fibonacci Zahlen
  - Kaskadierte Rekursion
  - Angeblich Kaninchenpopulation nach n Jahren, u. d. Annahme, dass Ein- und Zweijährige genau 1 Nachkommen pro Jahr zeugen.
  - Kaskadierte rekursive Funktion dank 2 Aufrufen von fib():

$$fib(n) = \begin{cases} 1, & \text{falls n} <= 1 \\ fib(n\text{-}2) + fib(n\text{-}1), & \text{sonst} \end{cases}$$

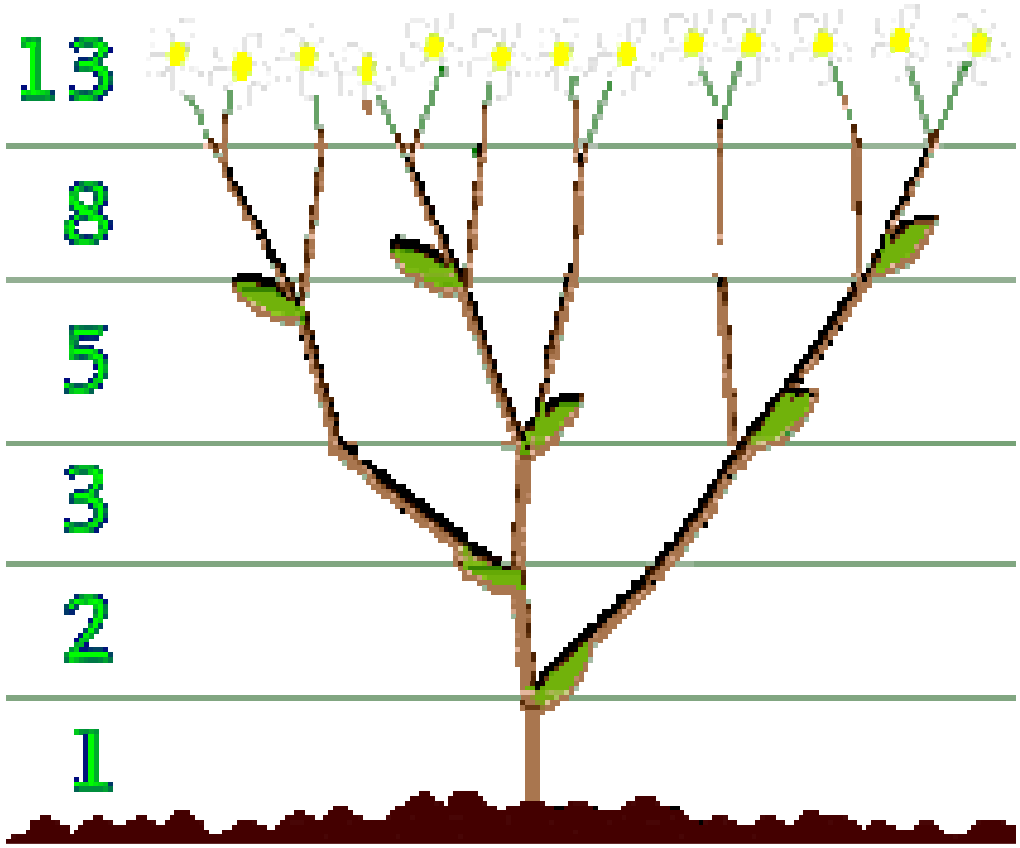
- Folge: 0, 1, 1, 2, 3, 5, 8, 13, …

technische universität dortmund

Dr. Lars Hildebrand

Achillea ptarmica (Sumpf-Schafgabe)

- Anordnung der Äste

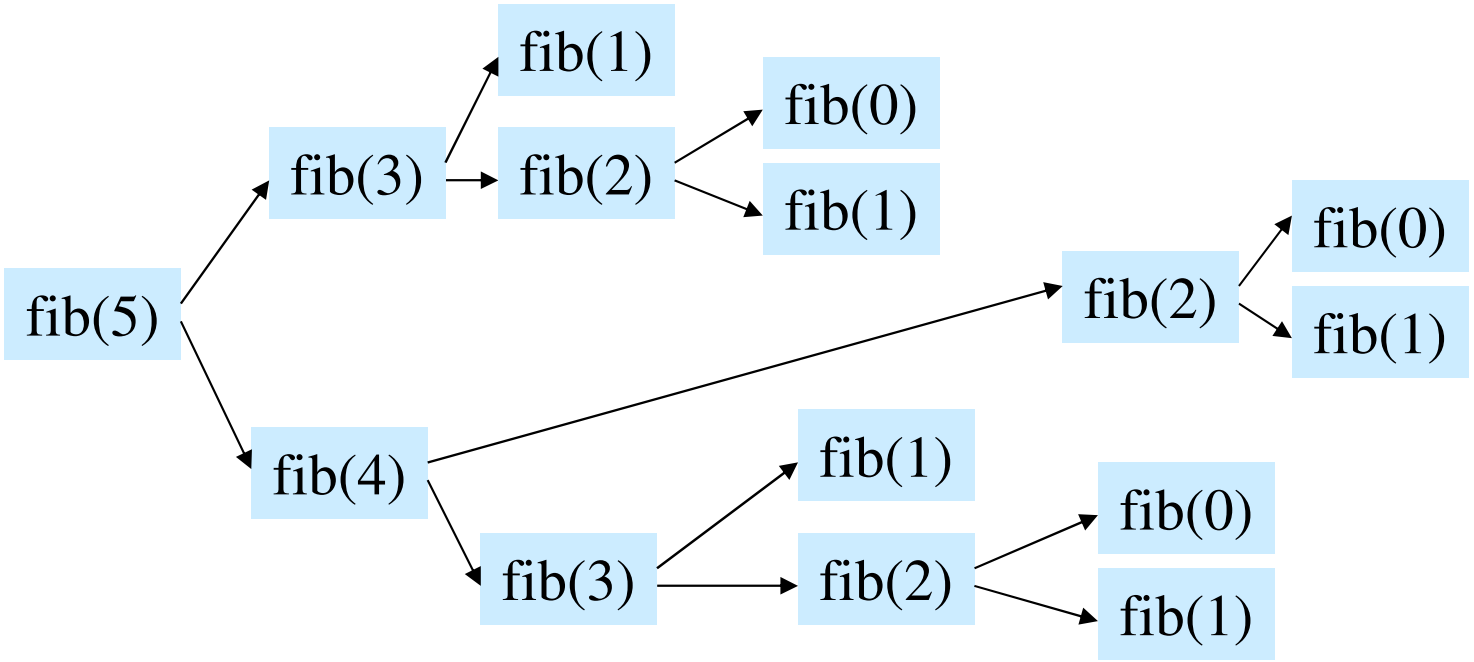


Quelle: http://www.goldennumber.net, Gary B. Meisner, 2006.

Fibonacci-Zahlen

- Beispiel fib(5)

$$fib(n) = \begin{cases} 1, & \text{falls n} <= 1 \\ fib(n\text{-}2) + fib(n\text{-}1), & \text{sonst} \end{cases}$$



- also: 15 Aufrufe der Funktion fib()

technische universität dortmund

Dr. Lars Hildebrand

Fibonacci-Zahlen und akkumulierende Parameter

- Noch ein **Trick zur Programmtransformation**: akkumulierende Parameter bewirken endrekursive Form
  - siehe auch Fakultätsfunktion
- **Beobachtung**:
  - In fib() werden viele Teilergebnisse mehrfach berechnet.
- **Generelle Technik**: Mehrfach verwendete Resultate sollten gespeichert werden, in Variablen, hier spezielle Parameter.

technische universität dortmund
Dr. Lars Hildebrand

Fibonacci-Zahlen und akkumulierende Parameter

- acc1, acc2 speichern Werte für fib(k-2), fib(k-1)

```
int fibo_acc(int n, int acc1, int acc2)
{
   if  (n=0) return(acc1) ;
   else
      return(fibo_acc(n-1,acc2,acc1+acc2));
}
int fib2(int n) { return(fibo_acc(n,1,1)) ; }
```

- **Bsp**: fib2(5) = fibo_acc(5,1,1) = fibo_acc(4,1,2) = fibo_acc(3,2,3) = fibo_acc(2,3,5) = fibo_acc(1,5,8) = fibo_acc(0,8,13)=8
- also 6 statt 15 rekursiver Funktionsaufrufe

Aspekte rekursiver Funktionen

- **Berechnungsaufwand**
  - entsteht durch die Stackverwaltung bei Aufruf / Terminierung einer Funktion
  - durch die Berechnung innerhalb der Funktion
- **Speicherplatzbedarf**
  - durch den einzelnen Funktionsaufruf auf dem Stack:
    - je Aufruf: alle lokalen Variablen und Aufrufparameter
  - Rekursionstiefe: Anzahl der Funktionsaufrufe, bis Rekursion endet

Aspekte rekursiver Funktionen

- **Terminierung**
  - erfordert Argumentation, warum Rekursionsanker tatsächlich erreicht wird
- **Korrektheit**
  - erfordert Argumentation, warum Resultat stimmt
  - bei rekursiven Problemstellungen oft formal über vollst. Induktion zu erbringen
- **Rekursion erlaubt oft sehr abstrakte und elegante Formulierung einer Lösung**
  - Denken Sie an Descartes rationale Mittel

Zusammenfassung

Dr. Lars Hildebrand

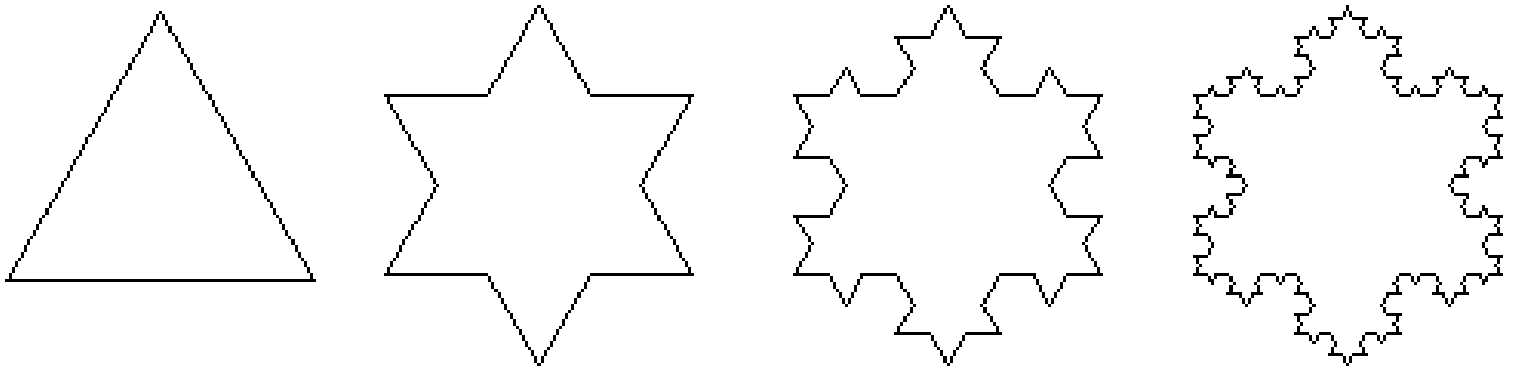
- **Unterprogramme**
  - Prozeduren, Funktionen, Methoden
  - Kommunikation zwischen Haupt- und Unterprogramm
    - call by value, call by reference
  - Top Down Entwurf: funktionale Zerlegung von Problemen

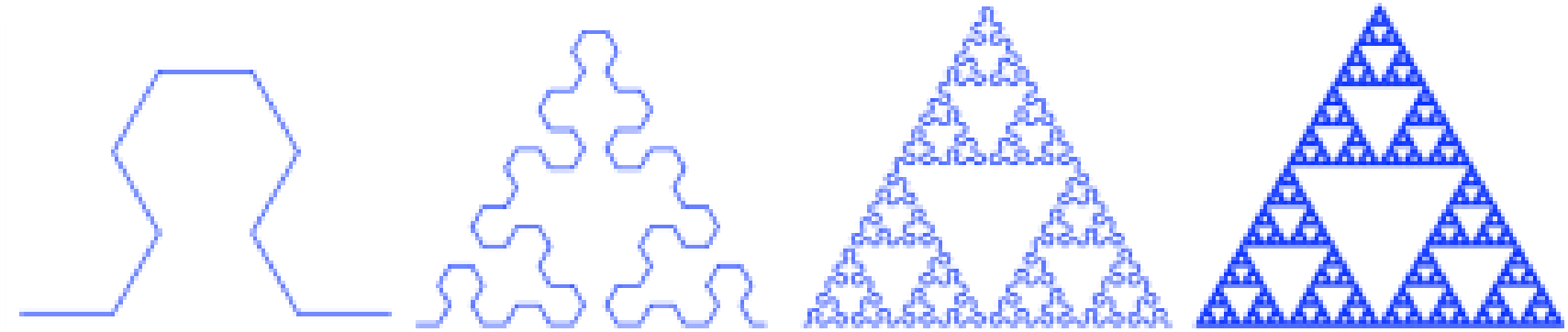
- **Rekursion**
  - primitiv rekursive Funktionen
  - endrekursive Funktionen
  - linear rekursive Funktionen
  - Ausführung rekursiver Funktionen: Aufrufstack
  - Transformation von rekursiven in iterative Funktionen
    - explizite Programmierung des Stacks für Parameter- und Zwischenwerte
    - Transformation endrekursiver Funktionen in iterative Funktionen
    - Transformation von rekursiven Funktionen in endrekursive Funktionen mittels akkumulierender Variablen

Etwas Entspannung

Dr. Lars Hildebrand

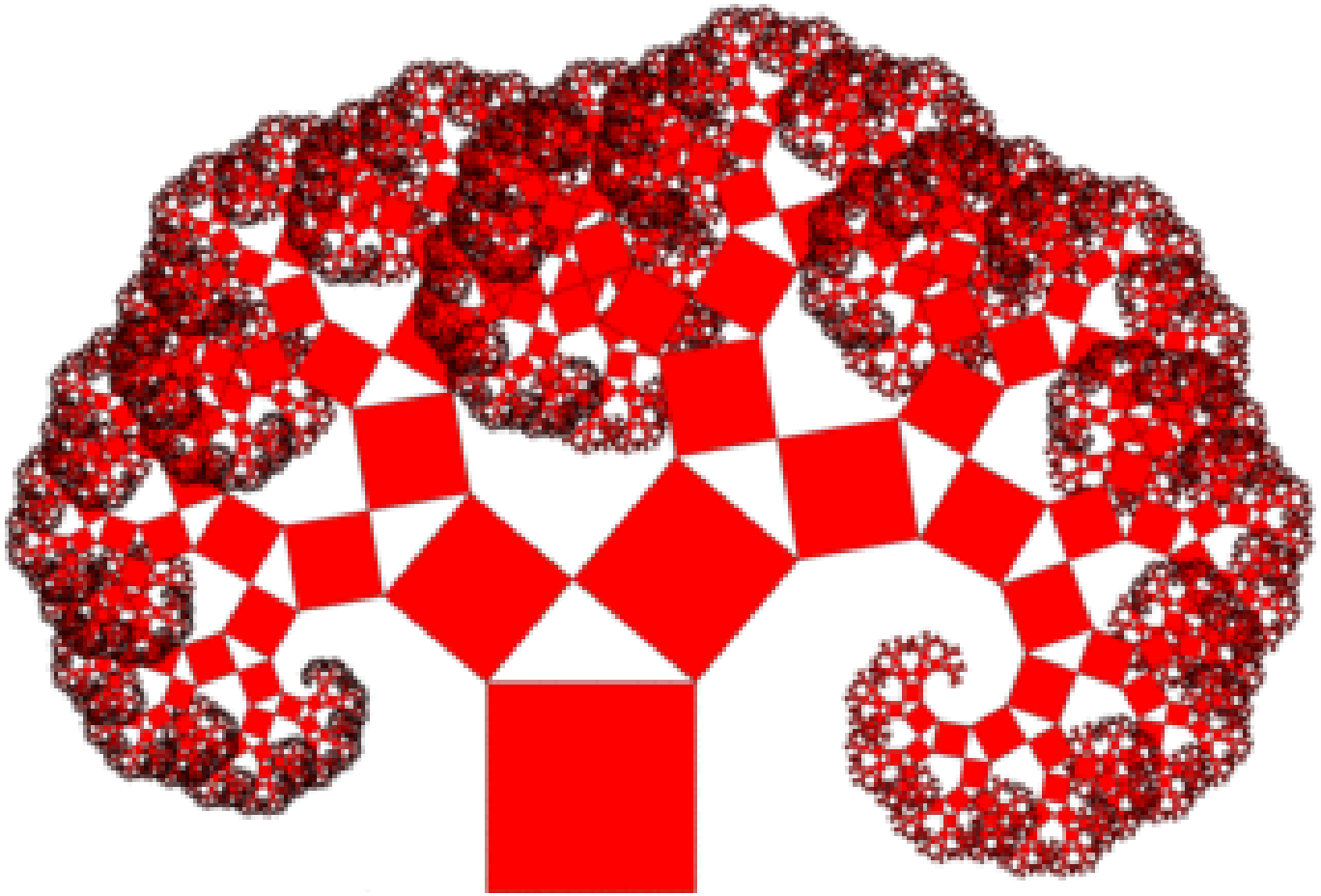
- Koch Kurven





Etwas Entspannung

- Pythagoras Baum



Quelle: http://www.4teachers.de, 2011.

Etwas Entspannung

- Blumenkohl

Quelle: http://wikipedia.de, 2011.

Etwas Entspannung

- Mandelbrot-Menge

Quelle: http://wikipedia.de, 2010, Yannick Gingras.

Vielen Dank für Ihre Aufmerksamkeit!

Nächste Termine

- Nächste Vorlesung

9.12.2011, 8:30
Campus Nord
EF50, HS 1