Teil 6: Debugging

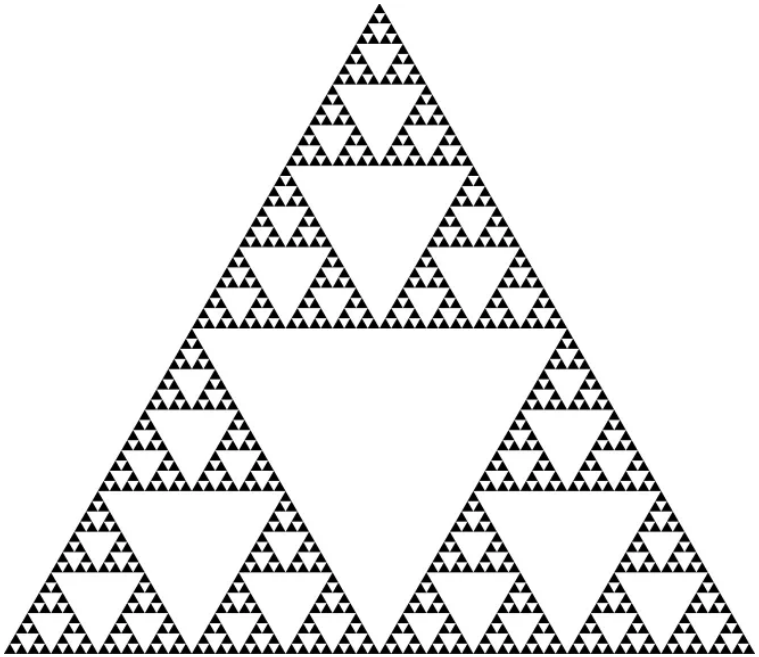
Gliederung

Rekursion

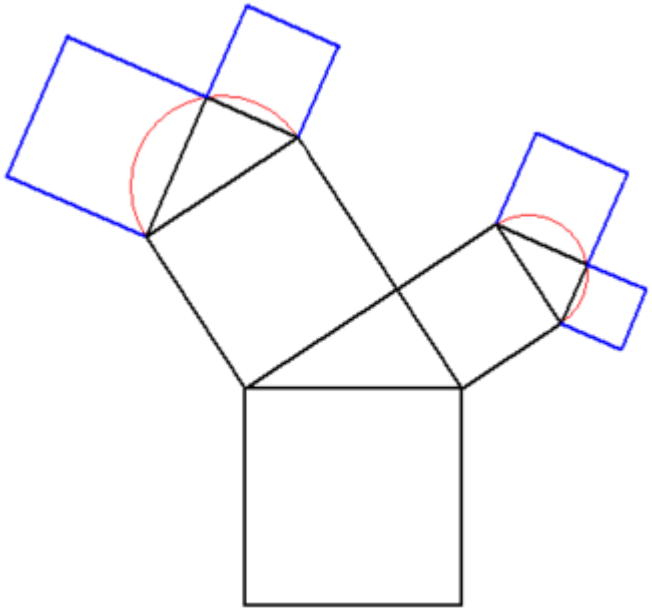
Debugging

Rekursive Funktionen

Beispiele

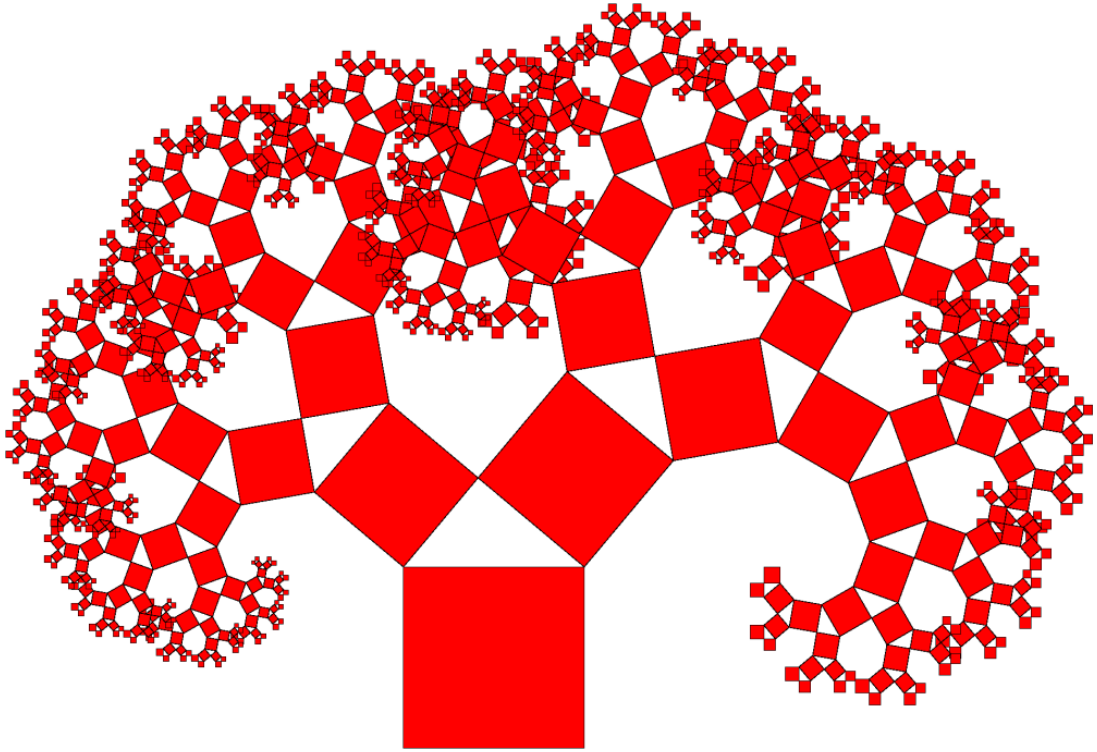


Sierpinski-Dreieck



Pythagoras-Baum

Beispiele



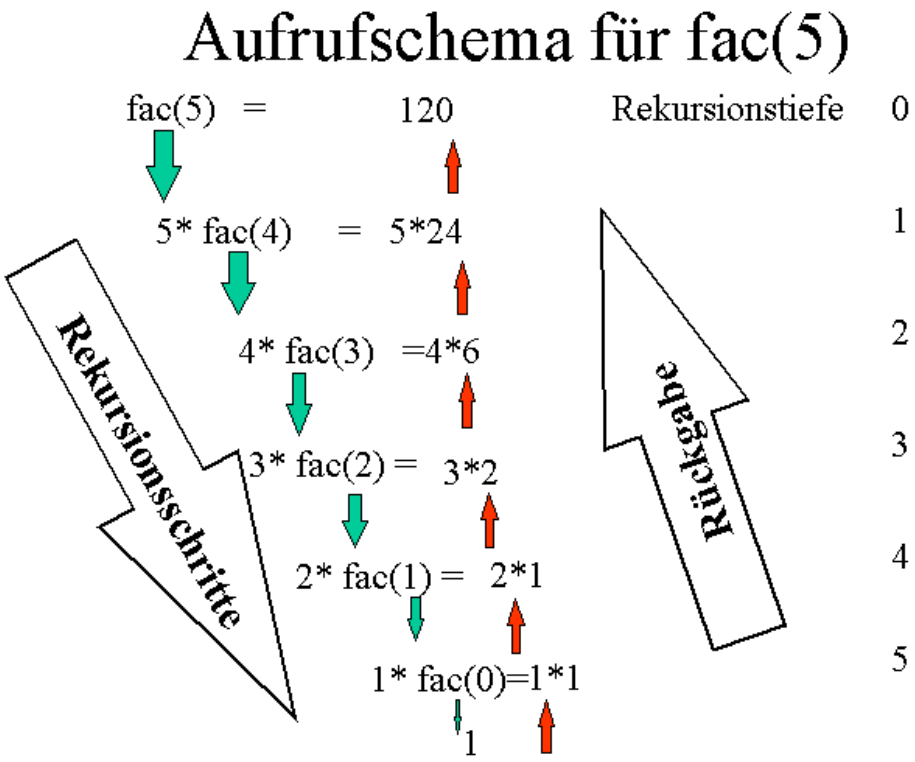
"ausgewachsener" Pythagoras-Baum

Rekursion als Problemlösungsstrategie

- Zurückführen einer "schwierigen" Aufgabe auf eine "einfachere" Aufgabe derselben Klasse
- Teile und Herrsche
- Prozeduren oder Funktionen können sich selbst aufrufen
- Rekursion vs. Iteration
- Jeder Aufruf der rekursiven Funktion muss sich durch Entfalten der rekursiven Definition in endlich vielen Schritten auflösen lassen.

Beispiel Fakultät

$$
\begin{array}{lll}
0! & = 1 & = 1 \\
1! & = 1 & = 1 \\
2! & = 1 \cdot 2 & = 2 \\
3! & = 1 \cdot 2 \cdot 3 & = 6 \\
4! & = 1 \cdot 2 \cdot 3 \cdot 4 & = 24 \\
5! & = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 & = 120 \\
 & \vdots &
\end{array}
$$



Debugging

- Fehler in einem Programm finden
- Code schrittweise "in Zeitlupe" ausführen
- Haltepunkte (Breakpoints)
- Speicherinhalte anzeigen
- Code benötigt Debug-Informationen
- Kommandozeilen-Debugger gdb

Beispiel

```
#include <stdio.h>

int main()
{
  int zahl = 0;
  char fehler;

  zahl++;
  zahl++;
  zahl++;
  zahl++;
  zahl++;
  zahl++;
  zahl++;
  zahl++;

  printf("Bitte Integer eingeben: ");
  scanf("%f",fehler);

  return 0;
}
```