



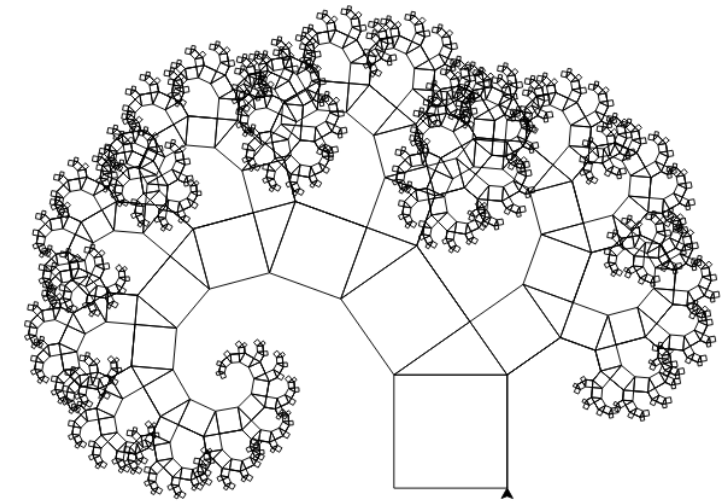
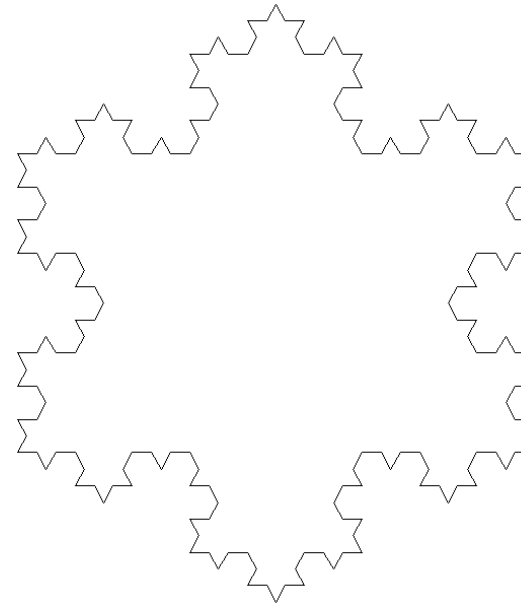
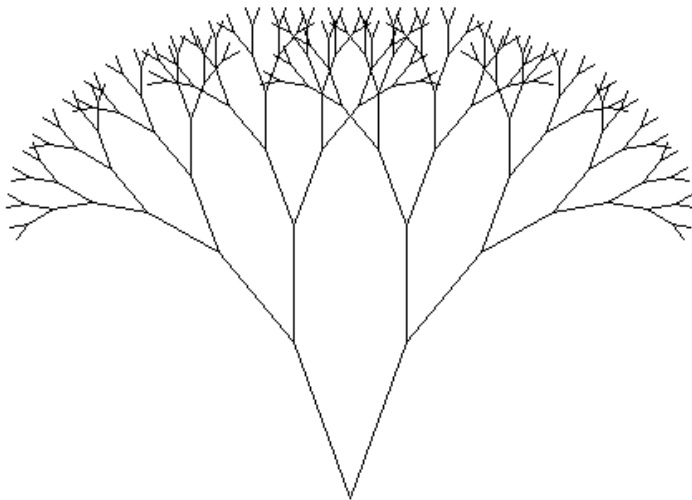
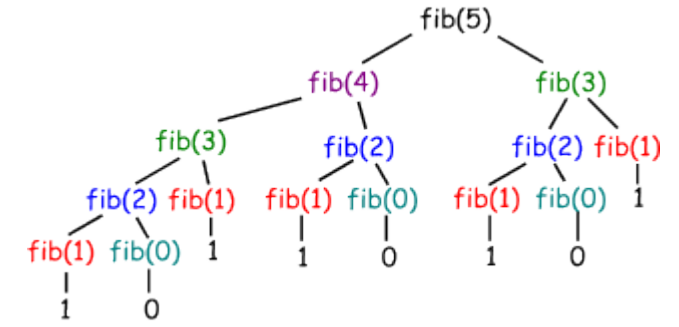
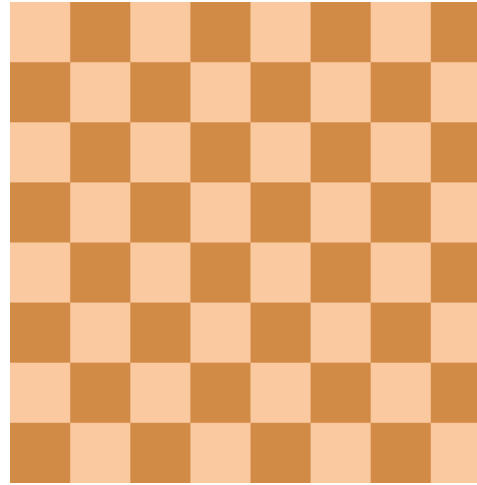
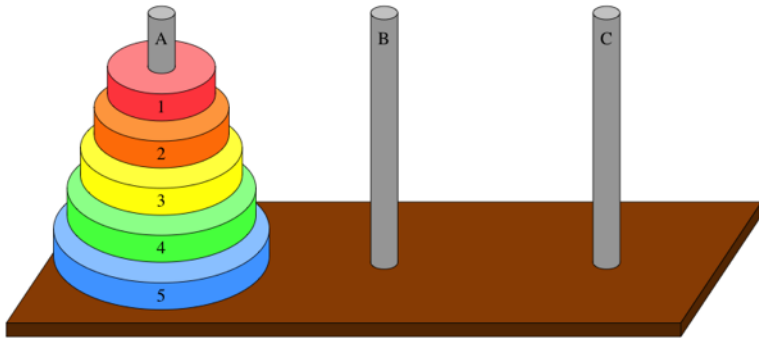
Rekursion



Rekursionsbeispiele

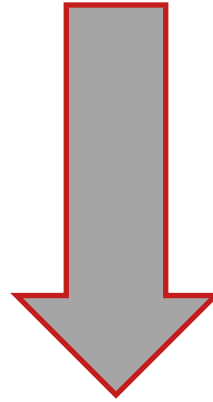
$$4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$$

$$5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$$



Rekursion als Problemlösestrategie

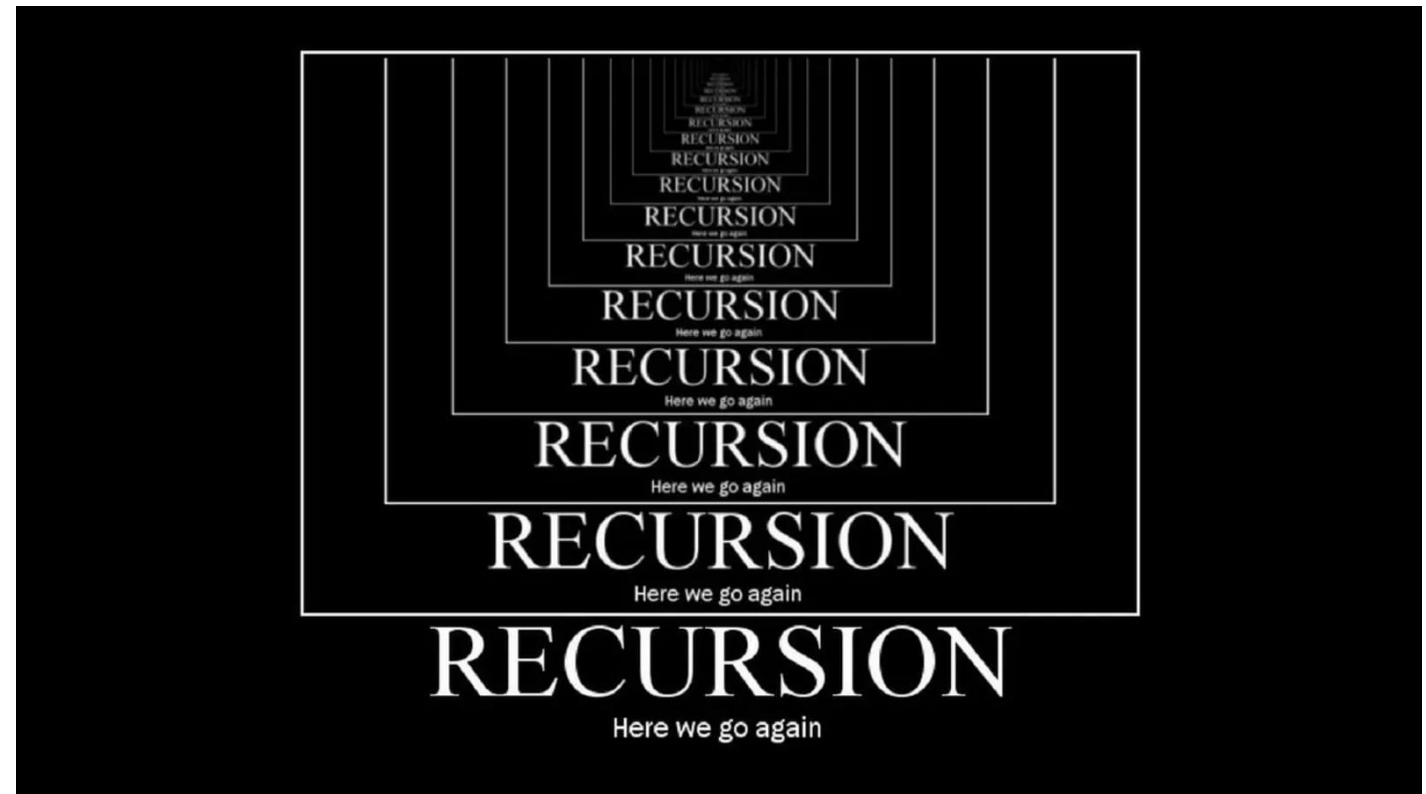
Was macht man mit einem großen Problem?



Man zerlegt es in kleinere, strukturgleiche Probleme!

Rekursion – wie funktioniert es?

- Rekursive Problemlösung:
- Schrittweise Lösung eines Teilproblems mit **gleicher Struktur**, aber **geringerer Funktionalität**.
- Eine rekursive Funktion verwendet bei der Lösung des Problems sich selbst – wie beim „Bild im Bild“:



Fakultätsfunktion

- Fakultät einer Zahl:

- $4! = 4 \cdot 3 \cdot 2 \cdot 1 = 24$

- $5! = 5 \cdot \underbrace{4 \cdot 3 \cdot 2 \cdot 1}_{= 4!} = 120$

- **Rekursion:** $5! = 5 \cdot 4! = 120$

- Rekursive Formel: $n! = n \cdot (n - 1)!$

- **Rekursionsschritt**

Fakultätsfunktion

- Rekursive Formel: $n! = n \cdot (n - 1)!$
- Rekursionsschritt
- Problem: Lösung von $1! = ?$

Fakultätsfunktion

- Rekursive Formel: $n! = n \cdot (n - 1)!, \text{ falls } n > 1$
 - Rekursionsschritt (Abbruchbedingung „umgekehrt“)
 - ~~Problem:~~ $1! = 1$
 - Rekursionsende
-
- **Lösung:** Für den Sonderfall $n = 1$ ist das Teilproblem direkt lösbar.
 - **Beachte:** Jede rekursive Funktion benötigt eine **Abbruchbedingung** sowie ein zugehöriges **Rekursionsende**.

Rekursion – Zusammenfassung

- Das **Prinzip der Rekursion** besteht darin, dasselbe Verfahren auf gleichartige, jedoch immer einfach werdende Problemsituationen anzuwenden (**Rekursionsschritt**).
- Jede rekursive Prozedur muss eine **Abbruchbedingung** enthalten, um das **Rekursionsende** – und damit eine Lösung – zu erreichen.

Fakultätsfunktion

- Die **Fakultät** einer Zahl:

$$5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$$

-

$$5! = 5 \cdot 4!$$

-

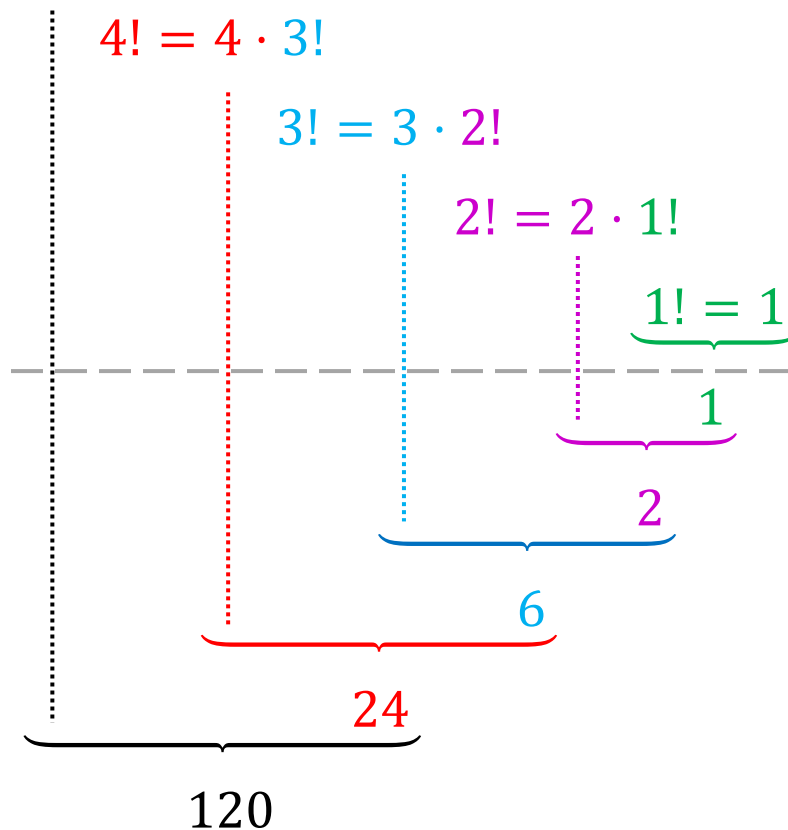
- Phase des
rekursiven **Abstiegs**
(rekursive Aufrufe)

-

- Phase des
rekursiven **Aufstiegs**
(Zusammenbau)

-

-



Fakultätsfunktion: die eigentliche Berechnung

- Die **Fakultät** einer Zahl:

$$5! = 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 120$$

-
- $5!$

$$= 5 \cdot 4!$$

- Phase des
rekursiven **Abstiegs**
(rekursive Aufrufe)
-

$$= 5 \cdot (4 \cdot 3!)$$

$$= 5 \cdot (4 \cdot (3 \cdot 2!))$$

$$= 5 \cdot (4 \cdot (3 \cdot (2 \cdot 1!)))$$

- Phase des
rekursiven **Aufstiegs**
(Zusammenbau)
-
-
-
-

$$= 5 \cdot (4 \cdot (3 \cdot (2 \cdot (1))))$$

$$= 5 \cdot (4 \cdot (3 \cdot (2)))$$

$$= 5 \cdot (4 \cdot (6))$$

$$= 5 \cdot (24)$$

$$= 120$$

Fakultätsfunktion: Pseudocode und Implementierung

- **Aufgabe:** Öffne den Ordner „Rekursion - Pseudocode“ auf dem Server bzw. die index.html-Datei.
 - a) Ordne die dargestellten Zeilen mittels Drag & Drop so, dass sie der Funktionalität der Methode **int fakultaet(int n)** entsprechen.
 - b) Öffne das BlueJ-Projekt „Übungen Rekursion Vorlage“ und implementiere die Methode aus der Teilaufgabe a).

Fakultätsfunktion: Lösung + iterative Variante

- Lösungsvorschlag:

```
• public int fakultaet(int n)
{
    if (n > 1)
    {
        return n * fakultaet(n-1);
    }
    else
    {
        return 1;
    }
}
```

Gegenüberstellung (iterativ):

```
public int fakultaetIterativ(int n)
{
    int erg = 1;
    while (n > 1)
    {
        erg = erg * n;
        n--;
    }
    return erg;
}
```

Wichtig: Rekursion ist eine Programmier Technik für eine Wiederholung. Sie ersetzt die Schleife durch das wiederholte Aufrufen der eigenen Methode.

Summe der Zahlen von 1 bis n

- Implementiere die Methode **summeBis**(**int** **n**) zur Berechnung der Summe von 1 bis n , das heißt:
- $1 + 2 + 3 + 4 + \dots + n$
- Nutze dabei eine **rekursive** Lösungsstrategie.

SummeBis: Lösung

- Lösungsvorschlag:

- ```
public int summeBis(int n)
{
 if (n > 0)
 {
 return n + summeBis(n-1);
 }
 else
 {
 return 0;
 }
}
```