Tagesprogramm

Fundiertheit und Fortschritt in der Rekursion

Formen der Rekursion

Rekursive Syntaxbeschreibungen

Simpel	::=	**a** { **a** }
Direkt	::=	Simpel | **(** Direkt **)**
IndirektA	::=	Simpel | **{** IndirektB **}**
IndirektB	::=	IndirektA **@** Simpel
Mehrfach	::=	Simpel | **(** Mehrfach ***** Mehrfach **)**
Eigen	::=	**[** Eigen **]**
Seltsam	::=	Seltsam

Fundiertheit und Fortschritt

Fundiertheit = Induktionsanfang = Abbruchbedingung

Fortschritt = Induktionsschritt = unterschiedliche Parameterwerte/Zustände

beides nötig, damit rekursive Definition Sinn ergibt

Endlosrekursion ohne Fundiertheit

```
public class Endlosrekursion1 {
    public static void main(String[] args) {
        seltsam(0);
    }

    private static void seltsam(int i) {
        seltsam(i + 1);
    }
}
```

Endlosrekursion ohne Fortschritt

```
public class Endlosrekursion2 {
    public static void main(String[] args) {
        seltsam(0);
    }

    private static void seltsam(int i) {
        if (i < 1000) {
            seltsam(i);
        }
    }
}
```

Direkte Rekursion

```
public class Direkt {
    public static void main(String[] args) {
        direkt(0);
    }

    private static void direkt(int i) {
        if (i < 1000) {
            direkt(i + 1);
        }
    }
}
```

Aufgabe: Vollständige Induktion und Fortschritt

Suchen Sie in Gruppen zu 2 bis 3 Personen eine Antwort auf folgende Frage:

Vollständige Induktion geht davon aus, dass $\mathbf{A(n)}$ für alle $\mathbf{n} \geq \mathbf{m}$ gilt.
Bei entsprechender Rekursion verringern wir $\mathbf{n}$ so lange, bis $\mathbf{m}$ (bzw. $\mathbf{m} - \mathbf{1}$ als Abbruchbedingung) erreicht ist.

Warum können wir die Richtung umdrehen und beginnend mit $\mathbf{0}$ den Wert so lange erhöhen, bis bei $\mathbf{1000}$ abgebrochen wird?

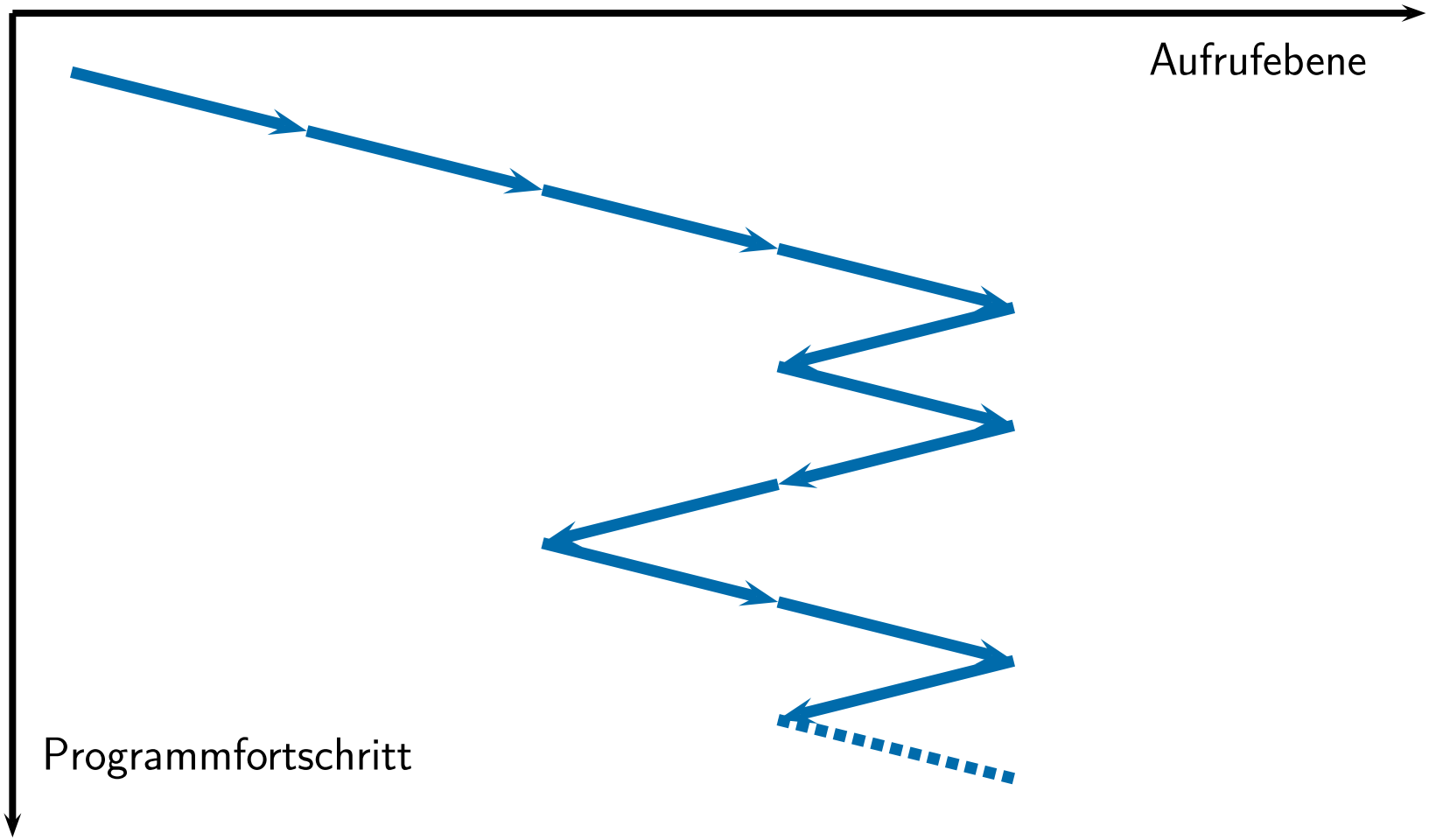
Indirekte Rekursion

```
public class Indirekt {
    public static void main(String[] args) {
        indirektA(0);
    }

    private static void indirektA(int i) {
        if (i < 1000) {
            indirektB(i);
        }
    }

    private static void indirektB(int i) {
        indirektA(i + 1);
    }
}
```

Mehrfach rekursive Aufrufe

```
public class TuermeVonHanoi {
    public static void main(String[] args) {
        rec(3, 'A', 'B', 'C');
    }

    private static void rec(int i, char a, char b, char c) {
        if (i > 0) {
            rec(i-1, a, c, b);
            System.out.println("von " + a + " nach " + c);
            rec(i-1, b, a, c);
        }
    }
}
```

Schematische Darstellung der Türme von Hanoi



Aufgabe: Nachvollziehen der Türme von Hanoi

Suchen Sie in Gruppen zu 2 bis 3 Personen in der Ausgabe von `TuermeVonHanoi` nach einer Systematik.

```
public class TuermeVonHanoi {
    public static void main(String[] args) {
        rec(3, 'A', 'B', 'C');
    }

    private static void rec(int i, char a, char b, char c) {
        if (i > 0) {
            rec(i-1, a, c, b);
            System.out.println("von " + a + " nach " + c);
            rec(i-1, b, a, c);
        }
    }
}
```