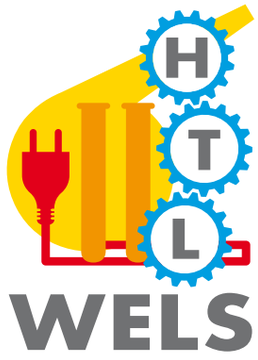
REKURSIONEN

SEW3

DI Thomas Helml



INHALT

- Definition
- Begriffe
- Vorteile/Nachteile
- Beispiel

DEFINITION

Rekursion ist die

„Definition eines Problems, einer Funktion oder eines Verfahrens durch sich selbst“

DEFINITION (SW-ENTWICKLUNG)

- Definition Rekursion
  - *Eine Funktion heißt rekursiv, wenn sie während ihrer Abarbeitung erneut aufgerufen wird*

- *Unterscheide*:
  - **Direkte Rekursion:**
    - Aufruf erfolgt in Funktionsrumpf der Funktion.
  - **Indirekte Rekursion:**
    - Aufruf erfolgt in einer anderen Funktion

BEGRIFFE

- Funktionsinkaration
  - konkreter Aufruf einer Funktion

- Rekursionstiefe
  - Anzahl der aktuellen Funktionsinkarnationen - 1

BEGRIFFE

- Iterativer Algorithmus
  - Algorithmus, der mit Schleifen arbeitet

- Rekursiver Algorithmus
  - Algorithmus, der rekursive Funkionen verwendet

REKURSIV VS. ITERATIV

- Satz:
  - zu **jedem rekursiv** formulierten **Algorithmus** gibt es einen adäquaten **iterativen Algorithmus**

VORTEILE VON REKURSIVEN ALGORITHMEN

- Vorteile
  - kürzere Formulierung
  - leichter verständliche Lösung
  - Einsparung von Variablen
  - teilweise sehr effiziente Problemlösungen (z.B. Quicksort)

NACHTEILE VON REKURSIVEN ALGORITHMEN

- **Nachteile**
  - weniger effizientes Laufzeitverhalten
    - Overhead beim Funktionsaufruf -> Stack!
  - Verständnisprobleme (nicht nur bei Programmieranfängern)
  - Konstruktion rekursiver Algorithmen ist „gewöhnungsbedürftig“

BEISPIEL FAKULTÄT – MATHEMATISCH

- Ein Beispiel einer Rekursion aus der Mathematik:
  - **Fakultät**

$$n! = \begin{cases} 1 & \text{falls } n = 1 \quad \textit{(Rekursionsanfang)} \\ n * (n-1)! & \textit{sonst} \quad \textit{(Rekursionsschritt)} \end{cases}$$

BEISPIEL – FAKULTÄT IN C

```
int fak(int n){
   if (n <= 1)
        return 1;
   else
        return n * fak(n-1);
}
```

BEISPIEL – FAKULTÄT REKURSIVE AUSFÜHRUNG

```
int fak(int n){
  if (n <= 1)
      return 1;
  else
      return n * fak(n-1);
}
```