Humboldt Universität Informatik

Grundlagen der Programmierung

Übung 6: Rekursion

Dr. Wolf Müller
Wolf.Mueller@informatik.hu-berlin.de

Rekursion - FAQ

- Was versteht man unter Rekursion?
  - **Lösung:** Prinzip der Rekursion ist es ein großes (schweres) Problem auf ein kleines (leichteres) Problem zurückzuführen. Der Grundgedanke hierbei ist, dass sich Funktionen selbst aufrufen.
- Was versteht man unter Restrekursion (tail-end recursion)?
  - **Lösung:** Rekursionen, deren rekursiver Schritt als allerletztes geschieht, dem also keinerlei weitere Berechnungen folgen. Vorteil: sehr einfach in Schleifen umwandelbar.
- Welchen Vor- und Nachteile haben rekursive Lösungen gegenüber Iterativen?
  - **Lösung:** Lesbarkeit und Wartbarkeit von rekursiven Lösungen ist höher. Iterative Lösungen sind hingegen i.d.R. schneller.

Rekursion - FAQ

- Lässt sich zu einer gegebenen rekursiven Lösung immer eine äquivalente iterative Lösung finden?
  - **Lösung:** Ja. Zum Bsp. kann ein in Haskell oder Prolog geschriebener Algorithmus in ein Fortran-Programm, welches Schleifen und GOTOs verwendet, umgeschrieben warden. (=>Turing-Vollständigkeit)

Restrekursion

- **Tail-end Recursion**
  - Rekursionen, deren rekursiver Schritt als allerletztes geschieht, dem also keinerlei weitere Berechnungen folgen.
  - Vorteil: sehr einfach in Schleifen umzuwandeln.
  - Schema:

```
func(n) {
    if (Abbruchbedingung) {
            return Wert(n)
    } else {
            m = berechne_einfacheres(n)
            return func(m) // Rekursion
    }
}
```

→

```
while (!Abbruchbedingung) {
    m = berechne_einfacheres(n)
}
return Wert(n)
```