Prinzipien zur Erstellung eines Programms

Imperativer Ansatz

Der Quellcode (formuliert in einer Programmiersprache, z. B. Pascal oder Python) besteht aus einer Folge von ausführbaren Anweisungen, die in der vorgegebenen Reihenfolge abgearbeitet werden.
In Maschinensprache (Assembler) geschriebene Programme verfolgen stets den imperativen Ansatz, die elementaren (Maschinen-)Befehle werden nacheinander ausgeführt.

Wesentliche Kontrollstruktur: **Iterationen** (for-, while-Schleife)

Funktionaler Ansatz

Der Quellcode bedient sich mathematischer Funktionen, durch die ein Algorithmus beschrieben wird.

Wesentliche Kontrollstruktur: **Rekursion**

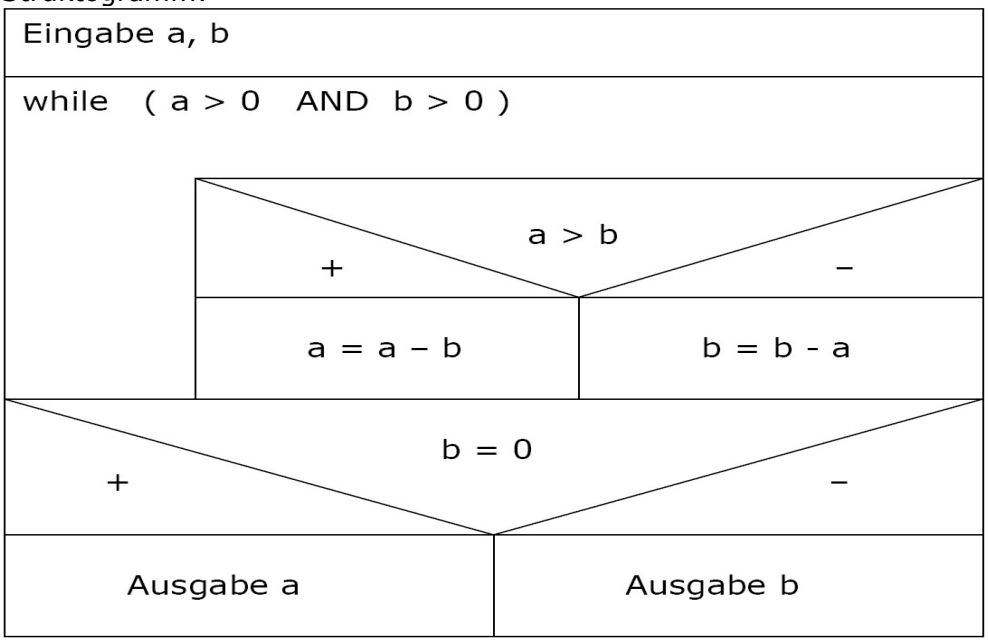
Definition:
*Eine Prozedur (Teilprogramm) oder eine Funktion heißt **rekursiv**, wenn ihr Anweisungsteil mindestens einen Aufruf von sich selbst enthält.*

Bei beiden Ansätzen ist durch eine Abbruchbedingung sicherzustellen, daß der Algorithmus terminiert, also nach endlich vielen Schritten beendet wird und zu einem Ergebnis führt.

Beispiel 1
Der Algorithmus **ggT** (**g**rößter **g**emeinsamer **T**eiler)

Nach Eingabe zweier natürlicher Zahlen a und b bestimmt ggT die größte ganze Zahl, durch die sich a und b jeweils ohne Rest teilen lassen.

a) **Imperativer Ansatz**, formuliert als **iterativer Algorithmus**
(„Euklidischer Algorithmus")
Struktogramm:



b) **Funktionaler Ansatz**, formuliert als **rekursiv definierte Funktion**

Die Funktion $(a, b) \rightarrow ggT(a,b)$ läßt sich rekursiv definieren:

Rekursionsanfang: $ggT(a,a) = a$

Rekursionsvorschrift: $ggT(a,b) = ggT(a-b, b)$, falls $a > b$

$ggT(a,b) = ggT(a, b-a)$, falls $b > a$

Aufgabe:
Realisiere den Algorithmus ggT als iteratives und als rekursives Python-Programm; vergleiche die Laufzeiten.

Beispiel 2
Die Funktion **„Fakultät"** (englisch: Factorial)

Die Funktion **fact** ordent jeder natürlichen Zahl **n** das Produkt
n! = 1 · 2 · . . . . . · n zu; definitionsgemäß gilt: 0! = 1.

a) **Imperativer Ansatz**

Formuliere den Algorithmus iterativ (for- oder while-Schleife) als Struktogramm und als Python-Programm

b) **Funktionaler Ansatz**

Die Funktion $n \rightarrow fact(n)$ läßt sich rekursiv definieren:

Rekursionsanfang: $fact(0) = 1$

Rekursionsvorschrift: $fact(n) = n \cdot fact(n-1)$, falls $n > 0$

Formuliere die Funktion **fact** als rekursives Python-Programm!

Beispiel 3
Die **Hofstadter-Funktion**

Die Funktion **hof** ist rekursiv definiert, $n \in \{1, 2, 3, ... \}$:

Rekursionsanfang: $hof(1) = 1$
$hof(2) = 1$

Rekursionsvorschrift: $hof(n) = hof(n - hof(n - 1)) + hof(n - hof(n - 2))$, $n>2$

Aufgabe:
Codiere den Algorithmus hofstadter
a) rekursiv,
b) iterativ
jeweils in Python; vergleiche insbesondere die Laufzeiten!

Hinweis zu b): Definiere in geeigneter Weise ein array (Feld), in dem bereits berechnete Funktionswerte gespeichert werden.