Arbeitsauftrag GK inf12 für 08.01.2021

Ergänzungen und Vertiefungen zum Konzept "Rekursion"

Die funktionale Formulierung eines Algorithmus bedient sich der mathematischen Struktur eines Problems; wesentliche Kontrollstruktur ist die Rekursion *(Bekanntlich heißt eine Funktion oder eine Prozedur (Teilprogramm) rekursiv, wenn sie mindestens einen Aufruf ihrer selbst enthält.)*.

In höheren Programmiersprachen (Pascal, C++, Python) ist die Möglichkeit der rekursiven Formulierung implementiert, was häufig eine sehr elegante Formulierung eines Algorithmus gestattet.

Bei rekursiven Programmen kann es jedoch zu einem „stack overflow" kommen, wenn die Anzahl der gleichzeitig aktiven Aufrufe der Prozedur oder der Funktion zu groß wird. In Python ist eine Rekursionstiefe von 1000 (oder 1024?) voreingestellt; nach Import des **sys-Moduls** mittels

```
import sys
```

läßt sich über

```
sys.getrecursionlimit()
```

die aktuelle Rekursionstiefe ausgeben, und mit

```
sys.setrecursionlimit(a)
```

kann man die Rekursionstiefe auf den Wert a setzen.

Wir greifen noch mal die Funktion fact (siehe Beispiel 3 aus Arbeitsauftrag für 24.11.2020) auf, die jeder natürlichen Zahl $n \in \{0, 1, 2, \ldots\}$ deren Falkultät fact(n) zuordnet.

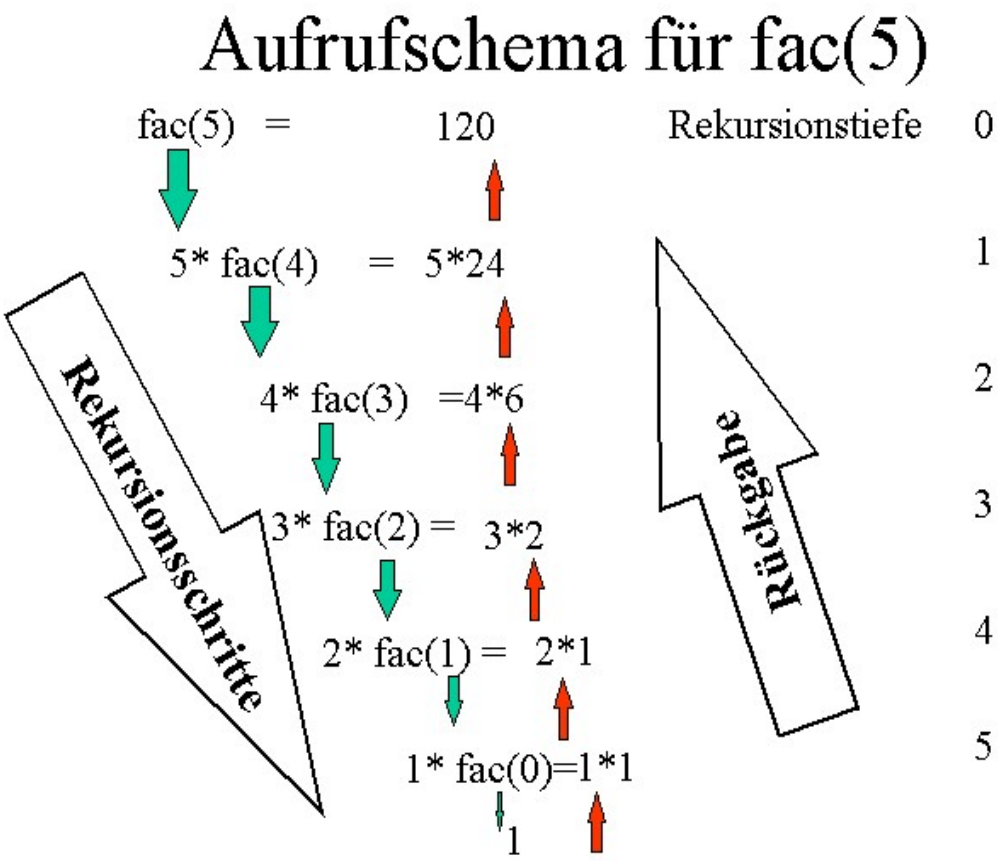
Python-Quelltext:

```
"factorial recursive"

import sys

print('Rekursionstiefe: ',sys.getrecursionlimit())
a=int(input('gewuenschte Rekursionstiefe: '))
sys.setrecursionlimit(a)
print('aktuelle Rekursionstiefe: ',sys.getrecursionlimit())
print()

def fact(x):
    global z
    z = z + 1
    if x == 0:
       return 1
    else:
       return x * fact(x - 1)

z = 0
n=int(input("n =  "))
y = fact(n)
print (n,"! = ",y)
print('Anzahl der Aufrufe: ',z)
```

Beachte:
Die Variablen **n**, **z** und **y** sind globale Variable; bei Aufruf **fact(n)** in der drittletzten Zeile wird der Wert der Variablen **n** an die lokale Variable **x** der Funktion **fact** übergeben (selbst wenn man innerhalb der Funktion **fact** die lokale Variable **x** mit **n** bezeichnen würde, würde für dieses **n** ein eigener lokaler Speicherplatz definiert).
Die globale Variable **z** zählt die Anzahl der Aufrufe der Funktion **fact**. Vor dem ersten Aufruf von **fact** wird **z** auf 0 gesetzt (initialisiert), und bei jeder Abarbeitung der Funktion **fact** wird **z** um 1 erhöht (inkrementiert). Die Anweisung **global z** verhindert, daß **z** innerhalb der Funktion als neue lokale Variable verstanden wird.

Aufrufschema für fact(5) (nach http://www.saar.de/~awa/jrekursion.html):

Der grüne Pfeil bedeutet jeweils „ruft auf", der rote „gibt zurück"; der Rekursionsanfang fact(0)=1 erzwingt, daß der Algorithmus abbricht (terminiert).



Aufgabe:
Erstelle zur Berechnung der **Hofstadter-Funktion** und der **Ackermann-Funktion** (Aufgabenblatt Nr. 2 vom 10.11.2020) jeweils einen Python-Programmtext, erweitert um die Möglichkeit, die Rekursionstiefe anzupassen und die Anzahl z der Aufrufe zu zählen und auszugeben; teste die Programme mit unterschiedlichen Werten.

Definition der **Hofstadter-Funktion**, die jeder natürlichen Zahl n≥1 den Wert hof(n) zuordnet:

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

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