

Der Call-Stack und die Rekursion

Ein populäres Beispiel für rekursive Algorithmen ist die Fakultätsfunktion:

```
5! = 5*4*3*2*1
fakultaet(5) = 120
fakultaet(3) = 3*2*1 = 6
```



(A1) Iterativ

Implementiere in BlueJ eine iterative Version der Fakultätsfunktion, die als Argument die Zahl entgegennimmt, deren Fakultät berechnet werden soll.



(A2) Rekursiv

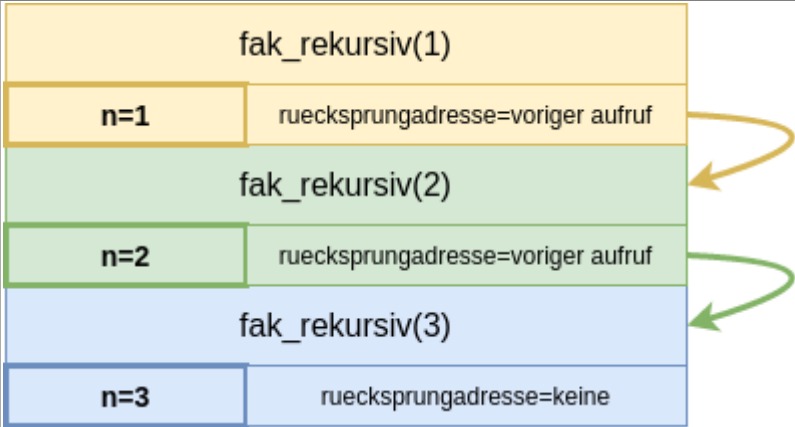
Implementiere anhand des folgenden Pseudocodes eine rekursive Version fak_rekursiv.

```
fak_rekursiv(int n):
  wenn n=1:
    return 1
  sonst:
    return n*fak_rekursiv(n-1)
```

- Was ist der Rekursionsfall, was der Basisfall?
- Teste deine rekursive Methode

Detaillierte Betrachtung des Call-Stacks bei der Rekursion

Was passiert	Wie sieht der Stack aus?

Was passiert	Wie sieht der Stack aus?
fak_rekursiv(3) wird aufgerufen. Auf dem Stack wird Speicher für diesen Aufruf reserviert. Es gibt keine Rücksprungadresse. Innerhalb dieses Aufrufs wird fak_rekursiv(2) (nächster Schritt) aufgerufen, da die Fallunterscheidung nicht zum Basisfall führt sondern zum Rekursionsfall.	 <pre> graph TD subgraph Stack direction TB f3[fak_rekursiv(3)] f3 --- n3[n=3] f3 --- ra3[ruecksprungadresse=keine] end </pre>
fak_rekursiv(2) wird aus dem vorhergehenden Aufruf heraus aufgerufen. Auf dem Stack wird Speicher für diesen Aufruf reserviert: Wichtig: Jeder Aufruf hat seinen eigenen Speicherbereich für Variablen, d.h. jeder Aufruf von fak_rekursiv hat sein eigenes n auf die die anderen Aufrufe nicht zugreifen können. Die Rücksprungadresse befindet sich jetzt im vorigen Aufruf der rekursiven Methode selbst. Das ist anders, als beim ersten Beispiel für den Call-Stack! Da erneut der Rekursionsfall eintritt, wird aus diesem Aufruf heraus fak_rekursiv(1) aufgerufen.	 <pre> graph TD subgraph Stack direction TB f2[fak_rekursiv(2)] f3[fak_rekursiv(3)] f2 --- n2[n=2] f2 --- ra2[ruecksprungadresse=voriger aufruf] f3 --- n3[n=3] f3 --- ra3[ruecksprungadresse=keine] end f2 --> f3 </pre>
fak_rekursiv(1) wird aus dem vorhergehenden Aufruf heraus aufgerufen. Auf dem Stack wird Speicher für diesen Aufruf reserviert. Die Rücksprungadresse befindet sich wiederum im vorigen Aufruf der rekursiven Methode selbst. Jetzt tritt der Basisfall ein!	 <pre> graph TD subgraph Stack direction TB f1[fak_rekursiv(1)] f2[fak_rekursiv(2)] f3[fak_rekursiv(3)] f1 --- n1[n=1] f1 --- ra1[ruecksprungadresse=voriger aufruf] f2 --- n2[n=2] f2 --- ra2[ruecksprungadresse=voriger aufruf] f3 --- n3[n=3] f3 --- ra3[ruecksprungadresse=keine] end f1 --> f2 f2 --> f3 </pre>

From: <https://info-bw.de/> -

Permanent link: https://info-bw.de/faecher:informatik:oberstufe:algorithmen:rekursion:callstack_rekursion:start?rev=1642074072

Last update: 13.01.2022 11:41

