

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

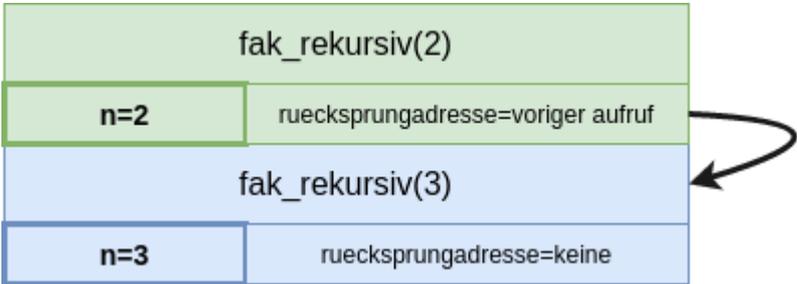
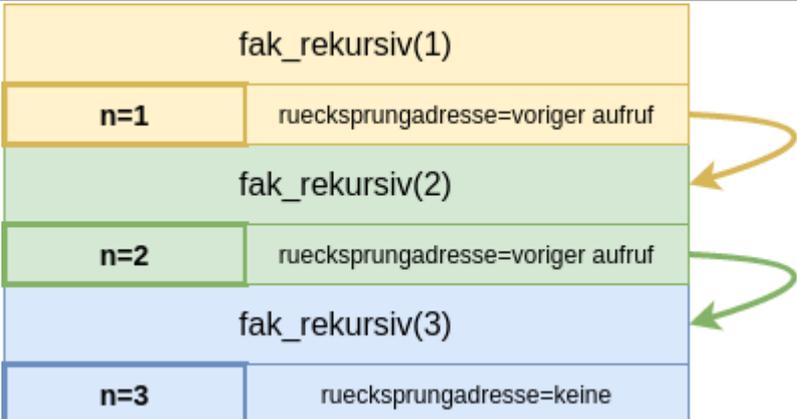
Implementiere anhand des folgenden Pseudocodes eine rekursive Version fak_rekursiv.

```
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?
<p>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.</p>	 <p>The diagram shows a single stack frame for 'fak_rekursiv(3)'. It is divided into two sections: 'n=3' and 'rücksprungadresse=keine'.</p>
<p>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.</p>	 <p>The diagram shows two stack frames. The top frame is 'fak_rekursiv(2)' with 'n=2' and 'rücksprungadresse=voriger aufruf'. Below it is 'fak_rekursiv(3)' with 'n=3' and 'rücksprungadresse=keine'. A curved arrow points from the 'rücksprungadresse=voriger aufruf' field of the 'fak_rekursiv(2)' frame down to the 'fak_rekursiv(3)' frame.</p>
<p>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!</p>	 <p>The diagram shows three stack frames. The top frame is 'fak_rekursiv(1)' with 'n=1' and 'rücksprungadresse=voriger aufruf'. Below it is 'fak_rekursiv(2)' with 'n=2' and 'rücksprungadresse=voriger aufruf'. At the bottom is 'fak_rekursiv(3)' with 'n=3' and 'rücksprungadresse=keine'. A curved arrow points from the 'rücksprungadresse=voriger aufruf' field of the 'fak_rekursiv(1)' frame down to the 'fak_rekursiv(2)' frame. Another curved arrow points from the 'rücksprungadresse=voriger aufruf' field of the 'fak_rekursiv(2)' frame down to the 'fak_rekursiv(3)' frame.</p>

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

Permanent link: https://info-bw.de/faecher:informatik:oberstufe:algorithmen:rekursion:callstack_rekursion:start?rev=1642073924

Last update: 13.01.2022 11:38

