

# 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

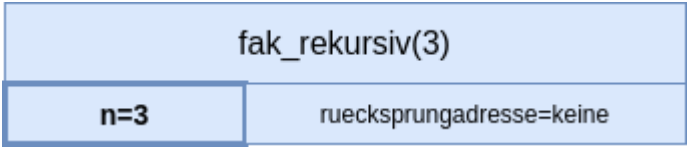
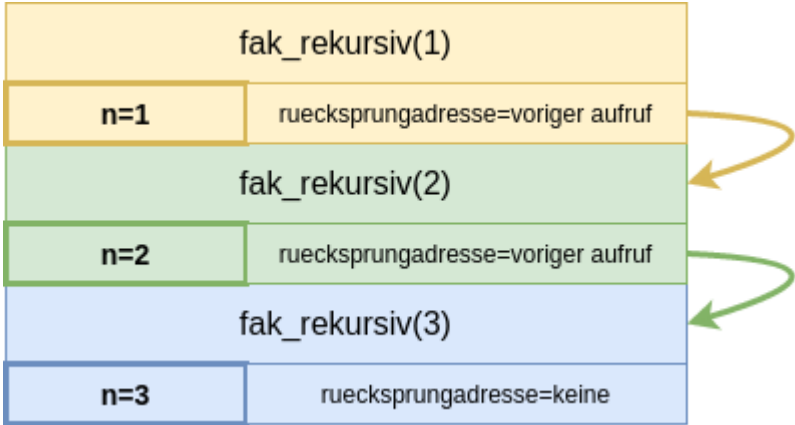
Implementiere anhand des folgenden Pseudocodes eine rekursive Version fak\_rekursiv.

```
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.</p> <p>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>fak_rekursiv(2) wird aus dem vorhergehenden Aufruf heraus aufgerufen.</p> <p>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 <b>im vorigen Aufruf</b> 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>fak_rekursiv(1) wird aus dem vorhergehenden Aufruf heraus aufgerufen.</p> <p>Auf dem Stack wird Speicher für diesen Aufruf reserviert. Die Rücksprungadresse befindet sich wiederum <b>im vorigen Aufruf</b> der rekursiven Methode selbst. <b>Jetzt tritt der Basisfall ein!</b></p>                                                                                                                                                                                                                                                                                                        |  |

| Was passiert                                                                                                                                                                                                                                                                                                    | Wie sieht der Stack aus?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Die ist der erste Aufruf, der vollständig abgeschlossen ist. Es wird kein neuer Aufruf von fak_rekursiv auf den Call-Stack gelegt, sondern der Aufruf von fak_rekursiv(1) endet damit, dass 1 an die aufrufende Stelle zurückgegeben wird und die zum Aufruf gehörenden Daten vom Stack entfernt werden.</p> | <p>The diagram illustrates the state of the call stack during a recursive process. The stack is shown as a vertical list of frames. The top frame, <code>fak_rekursiv(1)</code>, is grey and has a large red 'X' over it, indicating it is no longer active. Below it is the <code>fak_rekursiv(2)</code> frame in green, and at the bottom is the <code>fak_rekursiv(3)</code> frame in blue. Each frame contains a sub-frame with the current value of <code>n</code> and the return address (<code>ruecksprungadresse</code>). For <code>n=1</code>, the address is 'voriger aufruf'. For <code>n=2</code>, it is 'voriger aufruf'. For <code>n=3</code>, it is 'keine'. A yellow box labeled 'return 1' has an arrow pointing to the <code>n=1</code> frame, indicating the return value. A green arrow points from the <code>n=2</code> frame to the <code>n=3</code> frame, indicating the flow of execution.</p> |

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

Permanent link:  
[https://www.info-bw.de/faecher:informatik:oberstufe:algorithmen:rekursion:callstack\\_rekursion:start?rev=1642074537](https://www.info-bw.de/faecher:informatik:oberstufe:algorithmen:rekursion:callstack_rekursion:start?rev=1642074537)

Last update: 13.01.2022 11:48

