

Programmieren C: Rekursion, globale und statische Variablen: Rekursiver Binomialkoeffizient mit Cache

Klaus Kusche

Der Binomialkoeffizient “ n über k ” kommt in der Mathematik an vielen Stellen vor (siehe “Pascalsches Dreieck”). Er hat eine direkte Definition als $n! / (k! * (n-k)!)$ (deren Berechnung aber schnell zu Überläufen führt) und eine rekursive Definition, die sogar ohne Multiplikation auskommt:

- Ist $k > n$, so ist das Ergebnis 0.
- Ist $k == 0$ oder $k == n$, so ist das Ergebnis 1.
- Sonst ist das Ergebnis “ $(n-1)$ über $(k-1)$ ” + “ $(n-1)$ über k ”.

Aufgabe:

- Schreib eine Funktion mit zwei Parametern n und k , die den Binomialkoeffizient nach oben beschriebener Methode rekursiv berechnet und als Returnwert zurückliefert.
- Schreib dazu ein **main**, das zwei Zahlen von der Befehlszeile einliest und deren Binomialkoeffizient berechnet und ausgibt.
- Ergänze das um eine globale Variable, in der mitgezählt wird, wie oft die Funktion aufgerufen wird (erhöhe die Variable in der Funktion bei jedem Aufruf um 1), und gib die Gesamt-Anzahl der Aufrufe danach im Hauptprogramm aus.

Hinweise:

- Nimm auch hier **unsigned int** statt **int** für deine Variablen, Parameter und Returnwerte: Die obenstehende Definition gilt nur für n und k größergleich 0, und mit **unsigned int** sparst du dir die Prüfungen auf kleiner 0.
- Der Wertebereich von **unsigned int** kann für Binomialkoeffizienten ab n gleich 35 zu klein werden.

Zusatzaufgabe:

Bei höheren Werten (z.B. “30 über 15”) wächst die Anzahl der Aufrufe extrem, weil sehr viele Aufrufe mehrfach gemacht werden. Dem wollen wir mit folgender Idee abhelfen:

- Wir legen in der Funktion ein zweidimensionales statisches unsigned int-Array an. Der erste Index ist unser n , der zweite Index ist k . Eine Größe von $35 * 35$ reicht (prüfe in deiner Funktion, ob n kleiner 35 ist!), denn für höhere Werte von n und k würde “ n über k ” ohnehin überlaufen.
- Nach dem Prüfen der Rand- und Sonderfälle, aber vor den zwei rekursiven Aufrufen prüfen wir das Element $[n][k]$ des Arrays:
 - Ist es 0, so wurde der Binomialkoeffizient für dieses n und k noch nie berechnet. Wir berechnen ihn wie bisher mit 2 rekursiven Aufrufen, speichern das Ergebnis im Array, und geben es als Returnwert zurück.

- Ist es nicht 0, wurde dieser Wert schon einmal berechnet:
Wir geben den Wert aus dem Array sofort als Ergebnis zurück,
ohne nochmals rekursive Aufrufe zu machen.
- *Liefert das verbesserte Programm dasselbe Ergebnis?*
Um wieviel verbessert sich dadurch die Anzahl der Aufrufe?