Institut für Angewandte und
Numerische Mathematik

PD Dr. Mathias J. Krause
M.Sc. Stefan Karch
M.Sc. Mariia Sukhova 05.12.2022

Einstieg in die Informatik und Algorithmische Mathematik

Aufgabenblatt 8

Bearbeitungszeitraum: 19.12.2022 – 13.01.2023

Aufgabe 1 *Rekursive Berechnung der Fakultät*

Für natürliche Zahlen $n \in \mathbb{N}$ ist die Fakultät definiert durch

$$n! = 1 \cdot 2 \cdot \ldots \cdot n = \prod_{k=1}^{n} k, \tag{1}$$

also als das Produkt aller natürlichen Zahlen von 1 bis n. Rekursiv ausgedrückt ergibt sich

$$n! = \begin{cases} 1, & n = 0 \\ n \cdot (n-1)!, & n > 0. \end{cases} \tag{2}$$

Für negative Zahlen ist die Fakultät nicht definiert.
Erstellen Sie ein Java-Programm welches die Fakultät einer natürlichen Zahl rekursiv und nicht rekursiv berechnen kann. Gehen Sie dazu folgendermaßen vor:

- Erstellen Sie eine öffentliche Klasse `Fakultaet` und in dieser eine öffentlichen Methode `fac` mit einem ganzzahligen Rückgabewert. Diese Methode soll rekursiv die Fakultät einer der Methode übergebenen ganzzahligen Variablen `val` nach Gleichung (2) berechnen und zurückgeben.
- Erstellen Sie eine weitere Methode `facfor` gleichen Typs und mit gleichem Argument. In der Methode soll die Fakultät der Variablen `val`, ausgedrückt durch Gleichung (1) mithilfe einer **`for`**-Schleife berechnet und zurückgegeben werden.
- Erstellen Sie nun die `main`-Methode, in der zuerst eine ganzzahlige Variable `n` deklariert und gleichzeitig mit dem Wert 0 belegt wird. Lesen Sie dann einen Wert für die Variable `n` von der Konsole ein. Solange dieser Wert kleiner als 1 ist, soll ein neuer Wert für `n` eingelesen werden. Dies soll mithilfe einer **`while`**- Schleife überprüft werden. Falls ein $n \geq 1$ eingegeben wird, so soll zuerst die Methode `fac` mit dem Argument `n` aufgerufen und der berechnete Wert auf dem Bildschirm ausgegeben werden. Danach soll $n!$ mithilfe der Methode `facfor` berechnet und auch auf dem Bildschirm ausgegeben werden.

Musterlösung: Das Programm kann mit $n = 5$ mit dem Wert $5! = 120$.

Aufgabe 2 (Pflichtaufgabe) *Logarithmusberechnung*

In einem Java-Programm sollen zwei verschiedene Methoden zur näherungsweisen Berechnung von $\ln x$ verglichen werden. Es handelt sich dabei um eine rekursive Methode und ein Polynomapproximationsverfahren.

Verfahren 1 (Rekursion): Für einen reellen Wert $x > 1$ ergeben die mit der Rekursionsvorschrift

$$\begin{aligned} x_0 &:= x - 1, \\ z_0 &:= 1, \\ x_{n+1} &:= \frac{2 \cdot x_n}{1 + \sqrt{1 + z_n \cdot x_n}}, \\ z_{n+1} &:= \frac{z_n}{2}, \qquad\qquad n = 0, 1, 2, \ldots \end{aligned}$$

berechneten x_n eine monoton fallende Folge von Näherungswerten für $\ln x$, d. h. es gilt

$$x_0 > x_1 > x_2 > \ldots \qquad \text{mit } \lim_{n \to \infty} x_n = \ln x.$$

Verfahren 2 (Polynomapproximation): Gegeben ist das Polynom

$$p(t) = p_0 + p_1 \cdot t + p_2 \cdot t^2 + p_3 \cdot t^3$$

mit den Koeffizienten

$$p_0 = 2, \quad p_1 = 0.6704, \quad p_2 = 0.35370773 \quad \text{und} \quad p_3 = 0.48674609.$$

Eine Approximation für $\ln x$ lässt sich dann durch den Ausdruck $w \cdot p(w^2)$ mit $w = \frac{x-1}{x+1}$ berechnen.

Gehen Sie bei der Programmentwicklung folgendermaßen vor:

- Schreiben Sie eine Funktion `horner` mit einem `double`-Argument t und einem Argument `double[] p`, die den mittels des Hornerschemas berechneten Wert $p(t)$

  $$p(t) = (\ldots(p_n \cdot t + p_{n-1}) \cdot t + \ldots + p_1) \cdot t + p_0$$

  als Ergebnis liefert. Dabei ist n die obere Indexgrenze des Koeffizientenfeldes p.

  Hinweis: Verwenden Sie dazu eine **`for`**-Schleife!
- Schreiben Sie eine Funktion `ln_approx` mit `double`-Argument x, die den Wert $\ln x$ mit Hilfe von Verfahren 2 annähert. Vereinbaren Sie dazu innerhalb der Funktion das lokale Feld `double[] p` mit den Komponenten p_0, p_1, p_2 und p_3 wie angegebenen. Verwenden Sie dann die Funktion `horner` zur Berechnung von $p(w^2)$.
- Schreiben Sie eine Funktion `ln_rekurs` mit `double`-Argumenten x und z, die den Wert von $\ln x$ mit Hilfe von Verfahren 1 rekursiv berechnet. Dabei soll die Rekursion abgebrochen werden, wenn zwei auf der Maschine berechnete Näherungswerte x_k und x_{k+1} die Beziehung $x_k > x_{k+1}$ *nicht* mehr erfüllen. Die Rekursion soll nur die Berechnung der x_{n+1} (und somit auch die der z_{n+1}) beinhalten. In der `main`-Methode sollen später zuerst x_0 und z_0 berechnet und dann die rekursive Methode mit ihnen aufgerufen werden.

- Schreiben Sie ein Hauptprogramm, das zunächst einen x-Wert einliest und anschließend, solange $x \geq 1$ gilt, in einer Schleife
  - x_0 und z_0 berechnet,
  - die Näherungen l_r bzw. l_a für $\ln x$ durch die Funktionen `ln_rekurs` bzw. `ln_approx` berechnet,
  - die beiden relativen Fehler

    $$\delta_r = \frac{|\ln x - l_r|(x)|}{|\ln x|} \quad \text{und} \quad \delta_a = \frac{|\ln x - l_a|(x)|}{|\ln x|}$$

    berechnet,
  - die so berechneten Werte in der Form

    ```
    x = ...........   ln(x)   = ...........
                      l_r (x) = ...........   d_r = ...........
                      l_a (x) = ...........   d_a = ...........
    ```

    ausgibt und
  - einen neuen x-Wert einliest.

  Wird ein Wert $x < 1$ eingelesen, so soll das Programm mit einer entsprechenden Meldung beendet werden.
- Verwenden Sie zum Test Ihres Programms die Werte

  1.25, 1.316228, 1.5, 8.784306, 1.1435, 2.718281828, 3.16227, 4, und 0

  als Eingabedaten für x.
  Für $x = 2.718281828$ sollte die Ausgabe wie folgt aussehen:

  ```
  x = 2.718281828  ln(x)   = 0.9999999998311266
                   l_r (x) = 0.9999999998311271   d_r = 4.440892099250575E-16
                   l_a (x) = 1.0000384090332703   d_a = 3.840920215019398E-5
  ```

  Für $x = 4$ so:

  ```
  x = 4            ln(x)   = 1.3862943611198906
                   l_r (x) = 1.3862943611198915   d_r = 6.406853007629835E-16
                   l_a (x) = 1.385936488429824    d_a = 2.581505776143357E-4
  ```

Fragen 2 *Logarithmusberechnung*

- Beschreiben Sie kurz das rekursive Verfahren "Divide and conquer".
- Nennen Sie die vier Bestandteile aus denen ein Methodenkopf besteht und erklären Sie kurz derren Funktionen.

Aufgabe 3 *Fibonaccizahlen*

Die Fibonaccizahlen werden durch die folgende Vorschrift definiert:

$$F_1 := 1, \quad F_2 := 1, \quad F_n := a_{n-1} + a_{n-2}$$

Die Berechnung der N-ten Fibonaccizahl F_N kann also entweder dadurch erfolgen, indem man aufsteigend $F_3, F_4, ..., F_N$ berechnet, oder indem man rekursiv die Berechnung von F_N auf die Berechnung von F_{N-1} und F_{N-2} zurückführt. Schreiben Sie ein Java-Programm, welches die Berechnung in beiden Varianten implementiert. Gehen Sie dazu folgendermaßen vor:

(a) Erstellen Sie eine Klasse `Fibonacci` mit einer statischen Methode `it_fibo` vom Typ **`int`**. Diese soll als Argument eine Zahl n erhalten und die n-te Fibonacci-Zahl mittels einer **`for`**-Schleife aufsteigend berechnen und anschließend zurückgeben.

(b) Erstellen Sie eine Methode `rek_fibo` vom Typ **`int`**, welche ebenfalls zu einem Argument n die n-te Fibonaccizahl berechnet. Die Berechnung soll durch rekursive Aufrufe durchgeführt werden. Insbesondere soll keine Schleife verwendet werden.

(c) Erstellen Sie die `main`-Methode, in der ein Index n vom Benutzer eingelesen werden soll. Anschließend soll die n-te Fibonaccizahl mit den Methoden `it_fibo` und `rek_fibo` berechnet und jeweils ausgegeben werden.

Testen Sie die beiden Funktionen, indem Sie die 9-te, 10-te und 25-te Fibonaccizahl berechnen.