Institut für Angewandte und
Numerische Mathematik

PD Dr. Mathias J. Krause
M. Sc. Tino Lück
M. Sc. Joshua Doll

07.11.2024

Einstieg in die Informatik und Algorithmische Mathematik

Aufgabenblatt 4

Bearbeitungszeitraum: 24.11.2025 – 05.12.2025

Aufgabe 1 (Pflichtaufgabe) *Berechnung der Exponentialfunktion*

In dieser Aufgabe sehen Sie am Beispiel der Exponentialfunktion, dass es schon bei der Umsetzung einfacher mathematischer Formeln erhebliche Genauigkeitsprobleme geben kann.
Die Funktionswerte der Exponentialfunktion e^x können aus der Reihendarstellung

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \dots$$

berechnet werden. Die Funktionswerte sollen durch endliche Summen

$$e^x \approx S(N) := \sum_{i=0}^{N} \frac{x^i}{i!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \dots + \frac{x^N}{N!}$$

angenähert werden. Die einzelnen Summanden $y_i := x^i/i!$ lassen sich dabei wie folgt berechnen:

$$\begin{aligned} y_0 &:= 1 \\ y_i &:= \frac{x}{i} y_{i-1} \quad , \quad i = 1, 2, \dots \end{aligned}$$

Schreiben Sie ein Java-Programm, welches die Werte der Exponentialfunktion nach obigen Formeln berechnet. Legen Sie Ihren Berechnungen Variablen, welche Gleitkommazahlen speichern, zugrunde. Gehen Sie dazu folgendermaßen vor:

- Erstellen Sie eine öffentliche Klasse `Exponential` mit einer `main`-Methode. Hier soll der Benutzer zunächst dazu aufgefordert werden, die auszuwertende Stelle x sowie eine obere Summationsgrenze N einzugeben. Speichern Sie die Werte in entsprechende Variablen.
- Führen Sie die obige Summation zur Berechnung von $S(N)$ mittels einer **`for`**-Schleife durch, wobei N der eingegebenen Summationsobergrenze entspricht. Berechnen Sie dabei den Wert des aktuellen Summanden wie angegeben aus dem Wert des vorherigen Summanden. Definieren Sie sich dazu schon vor der **`for`**-Schleife eine Variable `summe`. Geben Sie das Ergebnis mit einem passenden Text auf der Konsole aus.

- Führen Sie die obige Summation erneut durch, jedoch soll dieses Mal die Summation erst abbrechen, wenn $|S(N) - S(N+1)| \leq 10^{-13}$ gilt. Nutzen Sie dafür eine **do-while**-Schleife. Bei der **do-while**-Schleife wird zur Überprüfung der Bedingung noch eine weitere Variable `summealt` benötigt. Geben Sie das Ergebnis, und bei welchem N das Ergebnis erreicht wurde, auf der Konsole aus.

  Hinweis: Die Betragsfunktion $|\cdot|$ heißt unter Java `Math.abs()`.

- Vergleichen Sie die berechneten Werte mit dem Ergebnis der Standardfunktion `Math.exp()`, welches Sie ebenfalls auf der Konsole ausgeben lassen sollen. Belegen Sie, dass dieses Verfahren insbesondere für negative x-Werte ungeeignet ist. Testen Sie Ihr Programm mit den Werten $x = \pm 1, \pm 10, \pm 50, \pm 100$.

- Verbessern Sie Ihr Programm, indem Sie die Berechnung mit automatischem Abbruch, nach der erfüllten Bedingung von oben, erneut implementieren, jedoch mit einer Änderung: Für negative x soll nun die Formel $e^{-x} = 1/e^x$ genutzt werden, d. h. berechnen Sie zunächst $e^{|x|}$ und invertieren Sie das Ergebnis wenn nötig. Geben Sie auch dieses Ergebnis mit erklärendem Text auf der Konsole aus.

- Testen Sie Ihr Programm erneut mit $x = \pm 1, \pm 10, \pm 50, \pm 100$. Lassen Sie sich hierbei die Ergebnisse aller jeweils verbesserten Versionen ausgeben und vergleichen Sie vor allem das verbesserte Ergebnis für negative Zahlen mit den Ergebnissen der Standardfunktion.

Musterlösung:

```
Bitte den Wert fuer x eingeben:
-100
Bis zu welchem Index soll die for-Schleife summieren?
1000
Der berechnete Wert nach 1000 Schritten: 8.144652745098073E25
Der berechnete Wert nach 243 Schritten mit Abbruch nach Genauigkeit: 8.144652745098073E25
Verbesserter Wert fuer negative x-Werte nach 192 Schritten: 3.7200759760208336E-44
Wert der Math-Bibliothek: 3.720075976020836E-44

Bitte den Wert fuer x eingeben:
50
Bis zu welchem Index soll die for-Schleife summieren?
100
Der berechnete Wert nach 100 Schritten: 5.184705527773213E21
Der berechnete Wert nach 118 Schritten mit Abbruch nach Genauigkeit: 5.184705528587081E21
Verbesserter Wert fuer negative x-Werte nach 118 Schritten: 5.184705528587081E21
Wert der Math-Bibliothek: 5.184705528587072E21
```

Fragen 1 *Berechnung der Exponentialfunktion*

- Welche Indizes besitzen das erste und das letzte Element eines Feldes in Java?
- Wieso ist das Verfahren für negative x-Werte ungeeignet? Erklären Sie.

Aufgabe 2 *Mittelwerte*

Erstellen Sie ein Java-Programm, das beliebig viele natürliche Zahlen ($a \in \mathbb{N}$) einliest und deren arithmetischen und geometrischen Mittelwert berechnet.
Das arithmetische Mittel A_N von N Zahlen $a_1, \ldots, a_N$ berechnet sich zu

$$A_N(a_1, \ldots, a_N) = \frac{1}{N} \sum_{i=1}^{N} a_i = \frac{1}{N}(a_1 + a_2 + \ldots a_N).$$

Das geometrische Mittel G_N von N positiven Zahlen $a_1, \ldots, a_N$ ergibt sich zu

$$G_N(a_1, \ldots, a_N) = \sqrt[N]{\prod_{i=1}^{N} a_i} = \sqrt[N]{a_1 \cdot a_2 \cdot \ldots \cdot a_N}.$$

Der Benutzer soll so lange Zahlen eingeben können, bis er die Eingabe durch eine Zahl $a \leq 0$ abbricht. Gehen Sie beim Erstellen des Programms folgendermaßen vor:

- Erstellen Sie eine öffentliche Klasse `Mittelwerte` mit einer `main`-Methode. Definieren Sie hier zunächst die ganzzahligen Variablen `N`, `summe` und `produkt` zum späteren Speichern der aktuellen Anzahl, der Summe und des Produkts der bisher eingegeben Zahlen.
- Fordern Sie den Benutzer mit entsprechendem Text dazu auf, die erste natürliche Zahl einzugeben und speichern Sie die Eingabe in einer passenden Variable.
- Schreiben Sie eine **`while`**-Schleife zur weiteren Berechnung und Eingabe. Die Schleifenbedingung soll überprüfen, ob die zuletzt eingegebene Zahl positiv ist. Ist dies erfüllt, so sollen innerhalb der Schleife die Variablen `N`, `summe` und `produkt` aktualisiert und die nächste Zahl eingelesen werden.

  Achten Sie darauf, dass die zu aktualisierenden Variablen vor Eintritt in die Schleife korrekt initialisiert werden (d. h. sinnvolle Anfangswerte besitzen).
- Berechnen Sie das arithmetische und geometrische Mittel aus der Summe bzw. dem Produkt der eingegebenen Werte.

  Hinweis: Allgemein erhalten Sie den Wert x^y durch die Methode `Math.pow(x,y)`.

Beispielergebnisse:

- $A_N(1, 2, 3, 4, 5, 6) = 3.5$
- $G_N(1, 2, 3, 4, 5, 6) \approx 2.994$

Hinweis: Sollten Sie für das arithmetische Mittel immer den Wert 0 erhalten, so könnte es sich lohnen, den Ausdruck `(1/N)` genauer zu betrachten. Warum erhält man unter Umständen einen anderen Wert als bei `(1.0/N)`?

Fragen 2 *Mittelwerte*

- Was ist der Unterschied zwischen `System.out.println()` und `System.out.print()`?

- Was ist der Unterschied zwischen einer **`while`**–Schleife und einer **`do`**–**`while`**–Schleife?

Aufgabe 3 *Integration mit Trapezregel*

Um den Wert eines bestimmten Integrals $I = \int_a^b f(x)\ dx$ näherungsweise zu bestimmen, genügt es, die Funktion f nur an einzelnen Punkten $x_0, \ldots, x_n$ (den sogenannten Stützstellen) auszuwerten. Eine Näherung für das Integral ist dann durch die Formel

$$I_n := \frac{h}{2}(f(x_0) + 2f(x_1) + \ldots + 2f(x_{n-1}) + f(x_n)) \quad \text{mit} \quad h := \frac{b-a}{n}$$

bestimmt, wobei die Stützstellen x_i durch

$$x_i := a + ih \ \text{ für } \ i = 0, 1, 2, \ldots, n$$

gegeben sind. Anschaulich approximiert man also die Fläche unter der Funktion durch n Trapeze mit den parallelen Seiten $f(x_i)$ und $f(x_{i+1})$ und der Höhe h.
Schreiben Sie ein Java-Programm, welches das Integral nach der Trapezregel berechnet. Erstellen Sie dazu eine öffentliche Klasse mit dem Namen `Trapezregel` und gehen Sie folgendermaßen vor:

- Definieren Sie eine statische Methode `f` mit einer Gleitkommazahl als Rückgabewert und einem Argument `x`, welches ebenso eine Gleitkommazahl speichert.

  Die Methode soll das Ergebnis des Ausdrucks

  $$f(x) := e^x \cdot \frac{1}{\sqrt{1+x^3}} \qquad (x \in \mathbb{R}, x > -1)$$

  zurückgeben, falls $x > -1$. Ist jedoch $x \leq -1$, so soll eine Fehlermeldung ausgegeben und das Programm mit dem Befehl `System.exit(1)` beendet werden. Nutzen Sie zur Berechnung von $f(x)$ die Methoden `Math.exp()` und `Math.sqrt()`.

- Definieren Sie eine statische Methode `Trapez`, welche eine Gleitkommazahl als Ergebnistyp besitzt und übergeben Sie als Argumente die beiden Intervallgrenzen a und b sowie die Schrittzahl n. `Trapez` soll den Näherungswert I_n gemäß der oben angegebenen Formel berechnen und zurückgeben.

- In der `main`-Methode sollen zunächst Gleitkommazahlen für die Integrationsgrenzen a und b deklariert und deren Werte dann vom Benutzer eingegeben werden. Bereiten Sie Ihr Programm entsprechend darauf vor Variablen einzulesen. In einer Schleife soll nun der Wert I_n mithilfe der Funktion `Trapez` in jedem Schritt neu berechnet werden. Beginnen Sie mit $n := 5$ und erhöhen Sie n in jedem Schritt um 5. Geben Sie die berechneten Werte nach jedem Schritt mit folgendem Text aus:

  ```
  Index n = ...  Naeherung In = ...
  ```

  Die Schleife soll abbrechen sobald $n > 100$ oder der *relative Fehler* klein genug ist:

  $$\frac{|I_{n-5} - I_n|}{|I_n|} < 10^{-6} \quad (I_0 := 0)$$

  Hinweis: `Math.abs()` liefert den Betrag einer Zahl.