

Übung 04: Gleitkommazahlen und Methoden

Abgabetermin: 14. 11. 2006

Name: _____

Matrikelnummer: _____

Gruppe: G1 (Prähofer) G2 (Prähofer) G3 (Wolfinger) G4 (Wolfinger)

Aufgabe	Punkte	gelöst	abzugeben schriftlich	abzugeben elektronisch	Korr.	Pkte
Aufgabe 04.1	12	<input type="checkbox"/>	Prosabeschreibung Ablaufdiagramm Java-Programm Testplan Testergebnisse	Java-Programm	<input type="checkbox"/>	
Aufgabe 04.2	12	<input checked="" type="checkbox"/>	Prosabeschreibung Ablaufdiagramm Java-Programm Testplan Testergebnisse	Java-Programm	<input type="checkbox"/>	

...

Aufgabe 04.2: Exponentialfunktion

Der Wert der Exponentialfunktion e^x für ein gegebenes x kann mit Hilfe der Taylorreihe

$$y(x) = e^x = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \frac{1}{5!}x^5 + \dots$$

angenähert werden.

Entwickeln Sie ein Java-Programm, das die Werte der Exponentialfunktion für alle Werte in einem variablen Wertebereich $x = [start .. end]$ an durch eine variable Schrittweite dx bestimmten Stellen mit einer variablen Fehlerschranke $epsilon$ berechnet und ausgibt.

Für jeden näherungsweise berechneten Funktionswert soll auch der „exakte“ Wert der Exponentialfunktion (mittels `Math.exp(x)`) und der Fehler $f(x) = y_{exakt}(x) - y_{genaehert}(x)$ ausgegeben werden, wohl wissend, dass man keinen wirklich exakten Funktionswert ermitteln kann.

Eine Sitzung soll folgendermaßen aussehen:

```

Bitte geben Sie den Startpunkt ein: 0
Bitte geben Sie den Endpunkt ein: 10
Bitte geben Sie die Schrittweite ein: 2
Bitte geben Sie die Fehlerschranke ein: 0.001
Zeile: x      exp(x) (genähert)      exp(x) (exakt)      Fehler
1:    0.0  1.0      1.0      0.0
2:    2.0  7.388994708994708  7.38905609893065  6.138993594273501E-5
3:    4.0  54.5978829056501  54.598150033144236  2.6712749413349E-4
4:    6.0  403.42863583131975  403.4287934927351  1.5766141535777933E-4
5:    8.0  2980.957677782414  2980.9579870417283  3.0925931423553266E-4
6:   10.0  22026.465632423035  22026.465794806718  1.6238368334597908E-4
    
```

Achten Sie besonders auf die Strukturierung des Programms in mehrere Methoden.

Hinweis: Die Berechnung des Näherungswertes soll abbrechen, sobald der Beitrag eines Terms zur Näherung kleiner als oder gleich der Fehlerschranke ϵ ist.

Gehen Sie bei der Lösung wie folgt vor:

1. Stellen Sie das Verfahren in Prosa dar.
2. Stellen Sie die wichtigste Methode (!) in einem Ablaufdiagramm dar.
3. Realisieren Sie das Verfahren in Java.
4. Führen Sie zur Überprüfung der Funktion sinnvolle Testausgaben ein.
5. Stellen Sie einen Testplan auf, d.h. stellen Sie interessante Testfälle auf und geben Sie die erwarteten Ergebnisse an.
6. Testen Sie Ihr Programm nach diesem Testplan.

Lösung

1) Prosabeschreibung

Hauptmethode

Einlesen des Startwertes start

Einlesen des Endwertes end

Einlesen der Schrittweite dx

Einlesen der Toleranz epsilon

Für alle Zahlen x von start bis end mit Schrittweite dx

Berechne den Näherungswert y von e^x mit Genauigkeit epsilon

Berechne den Wert mit der Methode `Math.exp(x)`

Berechne die Differenz von y und `Math.exp(x)`

Gib aus: Näherung y, Wert von `Math.exp(x)` und die Differenz

Funktion zu Näherung durch Taylorreihe mit Genauigkeit epsilon

Lösungsidee:

- In einer Schleife bilde die Reihe bis ein Glied der Reihe $< \epsilon$ ist
- In jeden Schleifendurchlauf soll ein neues Glied $1/i! * x^i$ gebildet und zur aktuellen Reihe addiert werden
- Beginne mit $i = 0$ und erstem Glied $1/0! * x^0 = 1$
- Verwende Variable i, iFac, xPower, term für das aktuelle Glied der Reihe und sum für die Reihensumme
- In jedem Schleifendurchlauf erhöhe i um eins; berechne neues iFac durch $iFac * i$ und neues xPower durch $xPower * x$

Ablauf

i = 0

xPower = 1

iFac = 1

term = 1

sum = term

Solange term $> \epsilon$

Erhöhe i um 1

bilde iFac = iFac * i

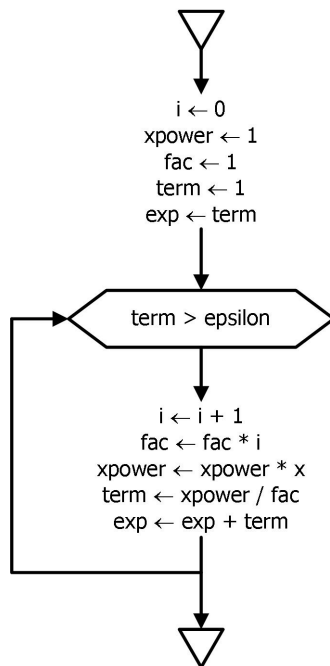
bilde xPower = xPower * x

term = 1 / iFac * xPower

sum = sum + term

Gib sum als Ergebnis zurück

2) Ablaufdiagramm

computeExponent(\uparrow p, \downarrow x, \downarrow epsilon)

3) Java-Programm

```

1 public final class Exponent {
2
3     static final boolean DEBUG_OUTPUT = false;
4
5     public static void main(String[] args) {
6
7         // Einlesen Startwert
8         double start = readStartValue();
9         // Einlesen Endwert
10        double end = readEndValue(start);
11        // Einlesen Schrittweite
12        double step = readStepValue(start, end);
13        // Einlesen Fehlerschranke
14        double eps = readEpsilon();
15
16        int i = 0;
17        Out.println("Zeile: x exp(x) (genaehert)   exp(x) (exakt)   Fehler");
18        // für jeden Schritt beginnend von start bis end
19        for (double x = start; x <= end; x = x + step) {
20            // Näherung berechnen
21            double appr = computeExponent(x, eps);
22            // exakten Wert berechnen
23            double exact = Math.exp(x);
24            // Differenz berechnen
25            double diff = Math.abs(appr - exact);
26            // Ausgabe der Zeile
27            i++;
28            Out.println(i + ":      " + x + "   " + appr + "   " +
29                exact + "   " + diff);
30        }
31    }
32
33    static double computeExponent(double x, double eps) {
34        double exp, term, xpower, fac;
35        int i;
36        // Initialisierung der einzelnen Werte für ersten Term (i = 0)
37        i = 0;
38        exp = term = fac = xpower = 1;
39        // Weitere Terme berechnen
40        while(term > eps) {
41            i++;
42            // x**i
43            xpower = xpower * x;
44            // i!
45            fac = fac * i;
46            // x**i / i!
  
```

```
47     term = xpower / fac;
48     // Summenbildung
49     exp = exp + term;
50
51     if(DEBUG_OUTPUT)
52         Out.println("i=" + i + ", xpower=" + xpower + ", fac=" + fac
53             + ", term=" + term + ", exp=" + exp);
54     }
55     return exp;
56 }
57
58 static double readStartValue() {
59     Out.print("Bitte geben Sie den Startpunkt ein: ");
60     double start = In.readDouble();
61     return start;
62 }
63
64 static double readEndValue(double start) {
65     Out.print("Bitte geben Sie den Endpunkt ein: ");
66     double end = In.readDouble();
67     while (start >= end) {
68         Out.println("Endpunkt <= Startpunkt! ");
69         Out.print("Bitte geben Sie einen gueltigen Endpunkt ein: ");
70         end = In.readDouble();
71     }
72     return end;
73 }
74
75 static double readStepValue(double start, double end) {
76     Out.print("Bitte geben Sie die Schrittweite ein: ");
77     double step = In.readDouble();
78     while (step >= end - start) {
79         Out.println("Schrittweite größer als Interval start .. end! ");
80         Out.print("Bitte geben Sie eine kleinere Schrittweite ein: ");
81         step = In.readDouble();
82     }
83     return step;
84 }
85
86 static double readEpsilon() {
87     Out.print("Bitte geben Sie die Fehlerschranke ein: ");
88     double eps = In.readDouble();
89     return eps;
90 }
91 }
```

4) Testausgaben

siehe Zeile 51-53 in 3) Java-Programm.

5) Testfälle

1. Fall

Startwert: 0
Endwert: 10
Schrittweite: 1
Epsilon: 0.001

2. Fall

Startwert: 0
Endwert: 10
Schrittweite: 1
Epsilon: 0.0000000001

3. Fall

Startwert: 10
Endwert: 100
Schrittweite: 10
Epsilon: 0.001

4.Fall

Startwert: 10
 Endwert: 100
 Schrittweite: 10
 Epsilon: 0.000000001

6) Test

Testfall #1

Bitte geben Sie den Startpunkt ein: 0
 Bitte geben Sie den Endpunkt ein: 10
 Bitte geben Sie die Schrittweite ein: 1
 Bitte geben Sie die Fehlerschranke ein: 0.001
 Zeile: x exp(x) (genaehert) exp(x) (exakt) Fehler

1:	0.0	1.0	1.0	0.0	
2:	1.0	2.7182539682539684	2.7182818284590455	2.7860205077168132E-5	
3:	2.0	7.388994708994708	7.38905609893065	6.138993594273501E-5	
4:	3.0	20.08546859390609	20.085536923187668	6.832928157862739E-5	
5:	4.0	54.5978829056501	54.598150033144236	2.6712749413349E-4	
6:	5.0	148.4129510721643	148.4131591025766	2.0803041229555674E-4	
7:	6.0	403.42863583131975	403.4287934927351	1.5766141535777933E-4	
8:	7.0	1096.6330406760985	1096.6331584284585	1.1775235998356948E-4	
9:	8.0	2980.957677782414	2980.9579870417283	3.0925931423553266E-4	
10:	9.0	8103.08370349121	8103.083927575384	2.2408417407859815E-4	
11:	10.0	22026.465632423035	22026.465794806718	1.6238368334597908E-4	

Testfall #2

Bitte geben Sie den Startpunkt ein: 0
 Bitte geben Sie den Endpunkt ein: 10
 Bitte geben Sie die Schrittweite ein: 1
 Bitte geben Sie die Fehlerschranke ein: 0.000000001
 Zeile: x exp(x) (genaehert) exp(x) (exakt) Fehler

1:	0.0	1.0	1.0	0.0	
2:	1.0	2.71828182845823	2.7182818284590455	8.15347789284715E-13	
3:	2.0	7.389056098925863	7.38905609893065	4.787281682183675E-12	
4:	3.0	20.085536923183504	20.085536923187668	4.163780431554187E-12	
5:	4.0	54.59815003313116	54.598150033144236	1.3073986337985843E-11	
6:	5.0	148.4131591025724	148.4131591025766	4.206412995699793E-12	
7:	6.0	403.42879349272846	403.4287934927351	6.650680006714538E-12	
8:	7.0	1096.6331584284492	1096.6331584284585	9.322320693172514E-12	
9:	8.0	2980.9579870417174	2980.9579870417283	1.0913936421275139E-11	
10:	9.0	8103.083927575374	8103.083927575384	1.000444171950221E-11	
11:	10.0	22026.465794806703	22026.465794806718	1.4551915228366852E-11	

Testfall #3

Bitte geben Sie den Startpunkt ein: 10
 Bitte geben Sie den Endpunkt ein: 100
 Bitte geben Sie die Schrittweite ein: 10
 Bitte geben Sie die Fehlerschranke ein: 0.001
 Zeile: x exp(x) (genaehert) exp(x) (exakt) Fehler

1:	10.0	22026.465632423035	22026.465794806718	1.6238368334597908E-4	
2:	20.0	4.8516519540958476E8	4.851651954097903E8	2.0551681518554688E-4	
3:	30.0	1.0686474581524465E13	1.0686474581524463E13	0.001953125	
4:	40.0	2.35385266837019968E17	2.35385266837019968E17	0.0	
5:	50.0	5.184705528587076E21	5.184705528587072E21	4194304.0	
6:	60.0	1.1420073898156842E26	1.1420073898156842E26	0.0	
7:	70.0	NaN	2.515438670919167E30	NaN	
8:	80.0	NaN	5.54062238439351E34	NaN	
9:	90.0	NaN	1.2204032943178408E39	NaN	
10:	100.0	NaN	2.6881171418161356E43	NaN	

Testfall #4

Bitte geben Sie den Startpunkt ein: 10
 Bitte geben Sie den Endpunkt ein: 100
 Bitte geben Sie die Schrittweite ein: 10
 Bitte geben Sie die Fehlerschranke ein: 0.000000001
 Zeile: x exp(x) (genaehert) exp(x) (exakt) Fehler

1:	10.0	22026.465794806703	22026.465794806718	1.4551915228366852E-11	
2:	20.0	4.8516519540979016E8	4.851651954097903E8	1.1920928955078125E-7	
3:	30.0	1.0686474581524465E13	1.0686474581524463E13	0.001953125	
4:	40.0	2.35385266837019968E17	2.35385266837019968E17	0.0	
5:	50.0	5.184705528587076E21	5.184705528587072E21	4194304.0	
6:	60.0	1.1420073898156842E26	1.1420073898156842E26	0.0	

7:	70.0	NaN	2.515438670919167E30	NaN
8:	80.0	NaN	5.54062238439351E34	NaN
9:	90.0	NaN	1.2204032943178408E39	NaN
10:	100.0	NaN	2.6881171418161356E43	NaN