Papieraufgaben

Kim Thuong Ngo

November 8, 2017

CONTENTS

1 BLATT 01

AUFGABE 1.2 Taschenrechner Papier

a) Wieso darf h bei der Berechnung der numerischen Ableitung in der Taschenrechneraufgabe nicht beliebig klein gewählt werden? Was passiert, wenn Sie h=0 setzen?

h darf bei der Berechnung nicht beliebig klein gewählt werden, da der Datentyp "double" eine Genauigkeit von 15 Signifikaten besitzt, d.h. es funktioniert nicht für sehr klein gewählte h.
Nach der Definition darf h nicht gleich Null gesetzt werden, da man nicht durch 0 teilen darf.

b) Was passiert bei der ggT Berechnung, wenn Sie eine negative Zahl eingeben und Sie diesen Fall nicht abfangen - also wenn Sie den iterativen bzw. rekursiven Algorithmus ganz normal ausführen?

Es entsteht eine Endlosschleife, d.h. die Funktion würde nicht terminieren und endlos weiterlaufen.

c) Führen Sie den ggT Algorithmus wie oben beschrieben auf ggT(42,63) und auf ggT(15,9) aus - geben Sie alle Unteraufrufe (bzw. rekursive Aufrufe) von ggT an!

ggT(42,63)
$\longrightarrow ggT(42,21)$
$\longrightarrow ggT(21,21)$
$\longrightarrow$ Lösung ist 21

ggT(15,9)
$\longrightarrow ggT(6,9)$
$\longrightarrow ggT(6,3)$
$\longrightarrow ggT(3,3)$
$\longrightarrow$ Lösung ist 3

AUFGABE 1.3 Induktionsbeweis
Beweisen Sie mittels vollständiger Induktion, dass 3 stets ein Teiler von $n^3 - n$ ist - für alle $n \in \mathbb{N}$

Induktionsanfang: $n = 1$
$n^3 - n = 1^3 - 1 = 1 - 1 = 0$
$\longrightarrow$ 0 ist durch 3 teilbar

Induktionsschritt: $n \longrightarrow n+1$
Induktionsvoraussetzung: $n^3 - n$ ist eine durch 3 teilbare Zahl

Induktionsbehauptung: $(n+1)^3-(n+1)$ ist durch 3 teilbar
Beweis:
$(n+1)^3-(n+1)$
$= n^3+3n^2+3n+1-n-1$
$= n^3+3n^2+2n$
$= n^3-n+3n^2+3n$
$= (n^3-n)+3(n^2+n)$
Nach IV ist n^3-n durch 3 teilbar.
Der zweite Term $3(n^2+n)$ ist ein ganzzahliges Vielfaches von 3, 3 ist durch 3 teilbar, und somit ist die Summer durch 3 teilbar.

□

2 BLATT 02

AUFGABE 2.2 Numerische Optimierung

a) Geben Sie die Stellen der gefundenen Minima (x,y) für die Werte $x_0 = 2,,6,8,10,12$ in einer Tabelle aus - und zwar für beide Funktionen und jeweils für $|x_l - x_{l+1}| < 10^{-7}$ und $|x_l - x_{l+1}| < 10^{-9}$

func=0: $f_1(x) = 1.84 + 1.42x - 2.4x^2 + 0.91x^3 + 0.124x^4 + 0.0055x^5$
func=1: $f_2(x) = (x-a)^2 + b$

Werte/Funktion	func=0 mit 10^{-7}	func=0 mit 10^{-9}
2	X=2.652110176077500 Y=0.287406447732254	X=2.6521884473720894 Y=0.2874064437749117
4	X=2.652268267405512 Y=0.287406447729125	X=2.652190027888739 Y=0.2874064437749115
6	X=9.35867462035928 Y=-5.53137102121286	X=9.35869082479512 Y=-5.53137102203488
8	X=9.35867460059416 Y=-5.53137102121087	X=9.35869082551920 Y=-5.53137102203493
10	X=9.35870737830612 Y=-5.53137102121104	X=9.35869115359424 Y=-5.53137102203510
12	X=9.35870731749117 Y=-5.53137102121696	X=9.35869115388596 Y=-5.53137102203505

Werte/Funktion	func=1 mit 10^{-7}	func=1 mit 10^{-9}
2	X=4.99994994521164 Y=-1.999999997494518	X= 4.99999949916960 Y=-1.999999999999749
4	X=4.99994992082009 Y=-1.999999997492076	X=4.999999499927895 Y=-1.99999999999975
6	X=5.00005007917991 Y=-1.999999997492076	X=5.00000050007210 Y=-1.99999999999975
8	X=5.00005005478834 Y=-1.999999997494518	X=5.0000005008304 Y=-1.999999999999749
10	X=5.00005007055525 Y=-1.999999997492939	X=5.00000050098819 Y=-1.999999999999749
12	X=5.00005007735475 Y=-1.999999997492258	X=5.00000050005405 Y=-1.99999999999975

b) Welche weiteren Abbruchbedingungen wären für die Anwendung an beliebigen Funktionen sinnvoll zu testen? Nennen Sie mind. zwei. Begründen Sie!

1.) Abbruch der Auswertung durch Überschreitung eines Wertes (definierte Grenze)
2.) Monotonie → streng monotone Funktion keine Extremstellen

AUFGABE 2.3 Schleifen, Reihen und Berechnungen

a) Sei $a_k = \frac{(-1)^k}{2k+1}$ das k-te Folgenglied einer Folge $(a_k)_{k=0,1,2,...}$. Die Folge beginnt demnach mit $1, \frac{-1}{3}, \frac{1}{5}, \frac{-1}{7}, ...$ Berechnen Sie Werte der Reihe der $4 * \Sigma_{k=0}^{n} a_k$. Benutzen Sie den Datentyp double. Geben Sie die Werte der Reihe für $n = 10^i$ mit $i = \{1,2,3,4,5,7\}$ aus. Gegen welchen Wert scheint die Reihe zu konvergieren?

	Werte
10^1	3.23231580940
10^2	3.15149340107
10^3	3.1425916543395
10^4	3.141692643590
10^5	3.1416026534897
10^6	3.1415936535887
10^7	3.1415927535898

→ die Werte konvergieren gegen π

b) Collatz-Vermutung
überprüfen der Vermutung für n=1,2,...10^6

Anzahl der Folgeglieder von x=[1000;1010] → x und 1 eingerechnet

	Anzahl
1000	112
1001	143
1002	112
1003	42
1004	68
1005	68
1006	68
1007	94
1008	112
1009	112
1010	63

Eingabe 10-19	Folgeglieder
10	5→16→8→4→2→1
11	34→17→52→26→13→40→20→10→5→16→8→4→2→1
12	6→3→10→5→16→8→4→2→1
13	40→20→10→5→16→8→4→2→1
14	7→22→11→34→17→52→26→13→40→20→10→5→16→8→4→2→1
15	46→23→70→35→106→53→160→80→40→20→10→5→16→8→4→2→1
16	8→4→2→1
17	52→26→1340→20→10→5→16→8→4→2→1
18	9→28→14→7→22→11→34→17→52→26→13→40→20→10→5→16→8→4→2→1
19	58→29→88→44→22→11→ 34→17→52→26→13→40→20→10→5→16→8→4→2→1

c) Auswertung eines Polynoms über ein Java-Programm:

$$f(x,y) = 333.75 * y^6 + x^2 * (11 * x^2 * y^2 - y^6 - 121 * y^4 - 2) + 5.5 * y^8 + \frac{x}{(2 * y)}$$

	Wert
Java-Wert - double	$-1,1805916207174113 * 10^{21}$
Java-Wert - float	$-6,338253 * 10^{29}$
Wolfram Alpha	-0.827396059946821368141165095479816291999033115784384819 91

Auswertung:
Der Wert von Wolfram Alpha ist am genauesten, da dieser ein Datentyp verwendet der mehr signifikante Stellen ermöglicht. Im Gegensatz dazu ist double mit seiner 64 Bit Darstellung zwar schlechter als Wolfram Alpha, aber wesentlich genauer als float.

3 BLATT 03

AUFGABE 3.2 Explizite Typumwandlung
Betrachten Sie die folgende Variablenzuweisung unter Java

int i = 5;
short s = 3;
float f = 5;
char a = 'a';
double d = 18.8;
long l = 34;
byte by = 1;

a) Welche der folgenden Zuweisungen sind zulässig? Korrigieren Sie die übrigen Anweisungen wenn möglich durch Einfügen einer expliziten Typveränderung

	Zulässigkeit	eventuelle Korrektur
l = 1;	√	
i = l + 90;	X	i = (int) l + 90;
d = f;	√	
f = s;	√	
by = d;	X	by = (byte) d;
a = i;	X	a = (char) i;
i = d;	X	i = (int) d;
f = i;	√	

AUFGABE 3.3 Rechnen mit Binärzahlen
Geben Sie für die im Zehnersystem dargestellten Zahlen x=63 und y=26 die zugehörigen Repräsentationen im Zweiersystem an.

	Zahl im Zehnersystem	Zahl im Binärsystem
x	63_{10}	111111_2
y	26_{10}	11010_2

Berechnung:

Division	Rest
63 / 2 = 31	1
31 / 2 = 15	1
15 / 2 = 7	1
7 / 2 = 3	1
3 / 2 = 1	1
1 / 2 = 0	1

Daraus folgt: $63_{10} \rightarrow 111111_2$

Division	Rest
26 / 2 = 13	0
13 / 2 = 6	1
6 / 2 = 3	0
3 / 2 = 1	1
1 / 2 = 0	1

Daraus folgt: $26_{10} \rightarrow 11010_2$

a) Addieren Sie die beiden Binärzahlen von Hand

Rechnung mit Übertrag in der Tabelle

$$\begin{array}{rr} & 111111 \\ + & 11010 \\ \hline & 1011001 \end{array}$$

Addition	Übertrag
1+0=1	0
1+1=0	1
1+0+1=0	1
1+1+1=1	1
1+1+1=1	1
1+1=0	1

b) Multiplizieren Sie die beiden Binärzahlen von Hand

$$\begin{array}{r} 111111 * 11010 \\ \hline 111111 \\ 111111 \\ 000000 \\ 111111 \\ 000000 \\ \hline 11001100110 \end{array}$$

Rechnung mit Übertrag in der Tabelle

$$\begin{array}{rr} & 1111110000 \\ & 111111000 \\ + & 1111110 \\ \hline & 11001100110 \end{array}$$

Addition	Übertrag
0+0+0=0	0
0+0+1=1	0
0+0+1=1	0
0+1+1=0	1
1+1+1+1=0	1+1
1+1+1+1+1=1	1+1
1+1+1+1+1=1	1+1
1+1+1+1=0	1+1
1+1+1+1=0	1+1
1+1+1+1=0	1+1

Überprüfung der Werte
Umwandlung der Zahlen von Binärsystem in Dezimalsystem und Vergleich mit Ergebnissen der Werte im Dezimalsystem von der Addition und Multiplikation

a) 1011001_2
$2^0 + 2^3 + 2^4 + 2^6$
$= 1 + 8 + 16 + 64$
$= 89_{10}$

$x + y = 26_{10} * 63_{10} = 89_{10}$ $\surd$

b) 11001100110_2
$2^1 + 2^2 + 2^5 + 2^6 + 2^9 + 2^{10}$
$= 2 + 4 + 32 + 64 + 512 + 1024$
$= 1638_{10}$

$x * y = 26_{10} * 63_{10} = 1638_{10}$ $\surd$

AUFGABE 3.4 Blöcke, Anweisungen und Schleifen
a) Kompilieren Sie folgende Programme und erläutern Sie die dabei auftretenden Fehler bzw. unerwartetes Verhalten.

i) Block 1:

```
class block1 {
  public static void main(String[] args) {
  int i; {
    i = 3;
    int j = 4
    }
```

```
  i = j;
  System.out.println(i);
  }
}
```

Fehlermeldung: j cannot be resolved to a variable
Erklärung:
Es gab ein Fehler bei der Übersetzung, weil die Deklaration der Variable j sich in einem inneren Block befindet, d.h. die Variable j ist nur innerhalb dieses Blockes gültig. Da sich die Wertzuweisung i=j außerhalb dieses Blockes befindet, kann dieser nicht auf die Variablendeklaration zugreifen und es entsteht die Fehlermeldung.

i) Block 2:

```
class block2 {
  public static void main(String[] args) {
  int i = 0; {
    i = 3;
    }
  System.out.println(i);
  }
}
```

Das Programm terminiert zum Wert 3.
Erklärung:
Die Deklaration der Variable i nimmt den Wert 0 an. Da aber, in einem inneren Block, eine neue Wertzuweisung für die Variable i eingeführt wird, ändert sich der Wert von 0 auf 3. Deshalb ist das Ergebnis des Programmes 3.

b) Welchen Wert hat die Variable x nach Ausführung der jeweiligen if-Anweisung

i) Anweisung A:

```
int x = 5;
if (x > 1 || x < 10) {
  x += 3;
  } else {
    x -= 2;
    }
```

$\rightarrow x = 8$

ii) Anweisung B:

```
int x = 1;
if (x>=1) {
  if (x<=1) {
    x = 7;
  }
}
```

$\rightarrow x = 7$

ii) Anweisung C:

```
int x = 10;
if (x>10) {
  if (x<10) {
    x = 1;
  }
} else {
  x = -1;
  }
```

$\rightarrow x = -1$

c) Welche Werte nimmt die Variable sum an, angefangen mit der Initialisierung über die Schleifeniterationen bis zum Ende der Ausführung der jeweiligen Schleife?

i) Schleife A:

```
int index = 8;
int sum = 0;
while(index >= 4) {
  index -= 2;
  sum += index;
  }
```

index	8	6	4	2
sum	0	6	10	12

ii) Schleife B:

```
int value = 6;
int sum = 0;
do {
  sum += value;
  value += 3;
}
```

```
while (value < 8);
```

value	6	9
sum	0	6

iii) Schleife C:

```
int value = 6;
int sum = 0;
for (\textbf{int} index = 0;$ index < 8$; index++) {
  sum += value;
  value += 3;
  index += 2;
  }
```

index	0	3	6	9
value	6	9	12	15
sum	0	6	15	27

4 BLATT 04

AUFGABE 4.3 Compiler-Fehler und Laufzeitfehler

a) Fragment 1

```
int[] a = new int[127];

for (int i = 0; i $\leq 127;$ i++ ) {
  a[i] = i;
}
```

Laufzeitfehler
Zeile 2 (**for**(**int***i* = 0; *i* ≤ 127; *i* + +) {) liefert das Problem.
Startbedingung der for-Schleife ist i ≤ 127, da aber das Array die Länge 127 hat, also das letzte Element des Arrays den Index 126 hat, kann die Schleife nicht laufen, weil sie erst ab Index 127 oder größer starten kann.

b) Fragment 2

```
int number = 303;
int[] b = \textbf{new} int[number];

while (number $>$ 0) {
  b[number$--$] = number + 7;
}
b[number] = -12;
```

Laufzeitfehler
Zeile 3 und 4 (**while** (number > 0) { und b[number−−] = number + 7;) liefert das Problem.
Es entsteht eine Endlosschleife in der while-Schleife, da number immer größer ist als 0, ist die Bedingung für die while-Schleife immer true, sodass sie ewig läuft.
c) Fragment 3

```
int[] c;
c = new int[12];

for ( $\textbf{int} i = c.length-1; i \geq 0; i-- $) {
  c[i] = "Hello World";
}
```

Compiler-Fehler
Zeile 4 (c[i] = "Hello World";) liefert das Problem.
Java liefert: "Type mismatch: cannot convert from String to int"

Der Datentyp des Arrays ist int, d.h. er kann seinen Datentyp nicht plötzlich auf String ändern. Daher kommt ein Compiler-Fehler, wenn man versucht einen String einem Index zu zuordnen.
d) Fragment 4

```
int num_fractions = 0;
float[] fractions = new float[num_fractions];

for(int i = 0; i < fractions.length; i++ ) {
  fractions[i] = 0.1 * i / fractions.length;
}
```

Compiler-Fehler
Zeile 4 (fractions[i] = 0.1 * i / fractions.length;) liefert das Problem.
Java liefert: "Type mismatch: cannot convert from double to float"
Der Datentyp des Arrays ist float. Da man aber in der Schleifenanweisung als Ergebnis, den Wert als double bekommt,also ein double einem Index vom Array float zuordnen will, gibt es ein Compiler-Fehlermeldung.

e) Fragment 5

```
char[] zeichen = new char[100];
int[] zahlen = new int[200];
for (int i = 0; i < zahlen.length; i++ ) {
  zeichen[i] = 'a';
  zahlen[i] = 1;
}
```

Laufzeitfehler
Zeile 3 (**for** (**int** i = 0; i < zahlen.length; i++) {}) liefert das Problem.
Die for-Schleife weist in ihren Anweisungen beide Arrays-Indizes Werten zu, da die Arrays aber nicht dieselbe Länge haben, gibt es Problem bei der Wertezuweisung ab Index 100 und es kommt zum Fehler.

f) Fragment 6

```
boolean[] true;
true[] = new boolean[2];
true[0] = false;
true[1] = true;
```

Compiler-Fehler
Alle Zeilen liefern den Fehler.
Nach boolean[] folgt der Variablenname des Arrays. Da der Bezeichner true ein reserviertes

Wort ist, kann man diesen nicht als Variablennamen verwenden. In den folgenden Zeilen ist es ein Folgefehler das true als Variablenname verwendet wurde.
Außerdem wurde in Zeile 2 das Array falsch erzeugt. Angenommen der Variablenname t wurde für das Ergebnis angegeben. So wäre die korrekte 2. Zeile entweder:
1. t = **new** boolean[2];
2. **boolean**[] t = **new** boolean[2];
Nach dem Variablennamen kommen keine eckigen Klammern!

G) Fragment 7

```
double Lichtg = 299792458.0; \\
double Lichtg2 = Lichtg * Lichtg; \\
double[] energy; \\

for ( int i = 1; i <= 10; i++ $) {
  double en = (double) i * Lichtg2;
  energy = new double[10];
  energy[i-1] = en;
}
System.out.println(energy[2]);
```

Compiler-Fehler
fehlende Zeile zwischen 3 und 4
Java liefert: "The local variable energy may not have been initialized"
Die Erzeugung des Arrays (bsp. energy = new double[100];) fehlt nach der Deklaration in der 3. Zeile. Da die for-Schleife aber auf das Array zugreifen will, aber zurzeit kein Array "existiert", entsteht eine Compiler Fehler an der Stelle.

h) Fragment 8

```
int[] a = new int[10];
int[] b = new int[10];
int k = 0;
for ($ k = 0; k \< 10; k++ $)
  a[k] = k;
b[k] = k;
if ( b[5] $<$ 10 ) {
  a[5] = b[6];
}
```

Laufzeitfehler
Zeile 6 (b[k] = k;) liefert das Problem.
Es fehlt eine Klammer um Zeile 5 und 6, sodass man die Anwendungen der for-Schleife zuordnen kann. Ohne Klammer wird nur Zeile 5 also "a[k] = k;" der for-Schleife zugeordnet

und es entsteht nur ein Wert im Array bei b[0]=0, d.h. die if-Abfrage kann gar nicht laufen, da keine Werte für den Index 5 und 6 im Array b existieren.

5 BLATT 05

AUFGABE 5.2 Realweltobjekte
Objekte repräsentieren Dinge aus der realen Welt.
Sie besitzen Eigenschaften (Attribute) und Funktionen (Methoden)

	Attribute	Methoden
Einkaufszettel	vat: double items[]: String price: double	calculatePrice(): double calculateVat(): double countItems(): integer
Getränk	color: String calories: double amountSugar: double	calculateVolume(): double tasteGood(): boolean freezingPoint(): double

AUFGABE 5.3 Verschiedene, klausurähnliche Aufgaben

a) Boolesche Ausdrücke und Präzedenzregeln.

$$x = 1.0f > 1.1f == -3 * +9 + 1/2 < -20 \text{ != true}$$

$x = 1.0f > 1.1f == -3 * +9 + 1/2 < -20$!= true
→ $x = 1.0f > 1.1f == (-3) * (+9)(+1/2) < -20$!= true
→ $x = 1.0f > 1.1f == -27 + 0.5 < -20$!= true
→ $x = 1.0f > 1.1f == -26.5 < -20$!= true
→ $x = false == true$!= true
→ $x = false$!= true
→ x = true

Welchen Wert hat x?
x = true

b)

i) Was bewirken die Anweisungen?

```
TurnierSpieler maier;
maier = new TurnierSpieler();
```

1) Variable vom Typ Turnierspieler deklarieren ohne einen Wert
2) der Variablen maier wird eine neue Instanz der Klasse TurnierSpieler ohne Parameter zugeordnet

ii) Schreiben Sie eine Codezeile, welche dem Spieler einen Namen zuweist.

```
maier.name = "Maier";
```

c)

i) Modifier für Attribute, bei einer strikten Kapselung
→ **private**

ii) Schleifenbedingung `while (this != null) { dosomething;}`, die in einer nicht-statischen Methode auftaucht, vereinfacht äquivalent darstellen
→ **while (true)** { **dosomething;**}

iii) Unterschied zwischen einer statischen und nicht-statischen Methode
→ **statische Methoden sind von "überall" aus sichtbar**
nicht-statische Methoden sind nur innerhalb der Instanz sichtbar

d) Geben Sie an, welche Ausgabe das folgenden Programm erzeugt

```
1.0
-1.0
-1.0
1.0
-1.0
1.0 (10.0)
```

E) Gegeben seien die Zahlen byte a = 31 und byte b = 15.

i) Wie werden a,b intern im Rechne dargestellt?
→ **a = 0001 1111**
b = 0000 1111

ii) Führen Sie mit der internen Darstellung die Rechnung c= a-b druch.
Gebe die Zwischenschritte an

Zweierkomplement: b = 1111 0001

```
    00011111
+   11110001
------------
    00010000
```

Addition	Übertrag
1+1=0	1
1+0+1=0	1
1+0+1=0	1
1+0+1=0	1
1+1+1=1	1
0+1+1=0	1
0+1+1=0	1
0+1+1=0	1

6 Blatt 06

7 Blatt 07

8 Blatt 08

9 Blatt 09

10 Blatt 010

11 Blatt 11