Autor = Schenk Philipp, 7093700

Aufgabe 1

Eigenschaften eines Algorithmus aus der Arbeit von Kaschube, Matthias und Matthäus, Franziska und Tolle, Karsten **[1]**:

- Ausführbarkeit (verständlich formuliert und ausführbar)
- Allgemeingültigkeit (nicht nur ein Einzelproblem)
- Determiniertheit (gleiche Startwerte entspricht gleichem Ergebnis)
- Determinismus (maximal eine Möglichkeit der Programmfortsetzung)
- Statische Finitheit (endlicher Quellcode)
- Dynamische Finitheit (Menge an Daten inklusive Zwischenspeicherungen sind endlich)
- Terminiertheit (kontrollierter Abbruch)

Gemäß dem Verfahren aus der Aufgabe:

- Ausführbarkeit: Wurde erfüllt. Die Beschreibung ist für den Leser verständlich, genau wie die Ausführung eines Algorithmus machbar ist.
- Allgemeingültigkeit: Größtenteils erfüllt. Ein Gegenbeispiel wäre ein Brötchen statt Toast, das nicht in den Toaster passen würde.
- Determiniertheit: Wurde erfüllt. Bei erneutem Durchlauf mit den gleichen Parametern (Toastzeit, Marmelade, …) entsteht das gleiche Ergebnis.
- Determinismus: Wurde erfüllt.
- Statische Finitheit: Wurde erfüllt.
- Dynamische Finitheit: Wurde generell erfüllt. Dennoch kann beispielsweise durch das Hängen der Toastertaste eine Endlosschleife entstehen.
- Terminiertheit: Wurde generell erfüllt. Dennoch kann beispielsweise durch einen leeren Brotbeutel ein unkontrollierter Abbruch auftreten, da kein Brot mehr vorhanden ist.

Aufgabe 2

Handlungsvorschrift 1:

- Terminierend, da sich b immer mehr (pro Durchlauf in der while-schleife) der 0 annähert. Dadurch, dass „Rest“ gerechnet wird und dieser dann h und später b entspricht pro Durchlauf wird h (und auch b) jedes Mal näher an 0 definiert und somit bricht es früher oder später kontrolliert ab.
- Determiniert, da gleiche Startwerte gleichem Ergebnis entsprechen. In diesen Rechnungen verändern sich bei gleichem Input auch die Rechnungen nicht und somit auch die Ergebnisse der Zwischenrechnungen. Keine Rechnungen mit Zufallswerten sind eingebaut.
- Deterministisch, da es eine fortlaufende Möglichkeit und keine Threads existieren. Bis die Bedingung erfüllt ist, läuft der Code in der while-schleife, danach geht es erst weiter.

Code: https://github.com/philippschenk2000/gpr

```
__author__ = "7093700, Schenk"
try:
    a = float(input('First number: '))
    b = float(input('Second number: '))
    while b != 0:
        h = a % b
        a = b
        b = h
        print(a)
except:
    print('Input has to be a number.')
```

Test 1:

```
First number: -20
Second number: 15.8
15.8
11.600000000000001
4.199999999999999
3.200000000000003
0.9999999999999964
0.2000000000000135
0.19999999999994245
7.105427357601002e-14
5.3290705182007514e-14
1.7763568394002505e-14

Process finished with exit code 0
```

Test 2:

```
First number: 0
Second number: -200.1
-200.1

Process finished with exit code 0
```

Test 3:

```
First number: 204
Second number: fnr
Input has to be a number.

Process finished with exit code 0
```

Test 4:

```
First number: 30.2
Second number: 2409
2409.0
30.2
23.200000000000056
6.999999999999943
2.2000000000002267
0.39999999999926317
0.20000000000391083
0.19999999999535234
8.558487252230407e-12
5.265121671982342e-12
3.2933655802480644e-12
1.971756091734278e-12
1.3216094885137863e-12
6.501466032204917e-13
2.1316282072803006e-14
1.0658141036401503e-14

Process finished with exit code 0
```

Handlungsvorschrift 2:

- Nicht terminierend: Der Algorithmus könnte rein technisch endlos laufen, für den Fall, dass a beispielsweise vor der while-schleife gleich 0 ist.
- Determiniert, da gleiche Startwerte gleichem Ergebnis entsprechen. Der Input ist nur indirekt zufällig, nämlich durch den Nutzer bestimmt. (Da wir nicht importieren dürfen, somit auch kein package „random“, somit gibt der Nutzer die Zahl an.) Wenn der Nutzer erneut diese Zahl eingibt, entseht durch die exakt gleichen Rechnungen auch der gleiche Ablauf uns somit das gleiche Ergebnis.
- Deterministisch, da es eine fortlaufende Möglichkeit und keine Threads existieren. Bis die Bedingung erfüllt ist, läuft der Code in der while-schleife oder rekursiv in einer Funktion, danach geht es erst weiter.

Aufgabe 3

a) Der Algorithmus fügt in eine Liste jeden ungeraden Wert zwischen 1 und (exklusive) 13, jedoch nicht die Zahl 5.
   Der Algorithmus ist deklarativ, da reine Fakten ausgedrückt werden / abgespeichert werden (ganzzahlige Hochzählung von 1 zu 13).
b) Der Algorithmus ist imperativ, da hierbei logische Schlussfolgerungen/ Wissen gefordert wird, wie die Liste [1, 3, …, 13] verläuft. Genau wie der Algorithmus aus a) fügt dieser alle Werte zwischen 1 und (exklusive) 13, jedoch nicht die Zahl 5 in eine Liste ein.

Literaturverzeichnis:

[1] Kaschube, Matthias und Matthäus, Franziska und Tolle, Karsten, *Grundlagen der Programmierung. Teil 2 – Algorithmus und Programm*. Frankfurt am Main: Goethe-Universität Frankfurt, 2023.