Übungen zu Computergrundlagen WS 2013/2014

Übungsblatt 9: Python, Gnuplot und Zahlensysteme

11.12.2013

Allgemeine Hinweise

- Abgabetermin für die Lösungen ist
  - **Dienstag, 17.12., 10:00** für die Übungsgruppen am Donnerstag und Freitag
  - **Mittwoch, 18.12., 10:00** für die Übungsgruppen am Montag und Dienstag
- Schickt die Lösungen bitte per Email an Euren Tutor.

Aufgabe 9.1: Das Heronverfahren und Gnuplot (5 Punkte)

Hat man keinen Taschenrechner zur Hand, liefert das mehr als 3000 Jahre alte Heronverfahren gute Nährungen für die Wurzel. Auf speziellen Prozessoren wie Grafikkarten wird das Verfahren sogar heute noch eingesetzt, um die Genauigkeit von Wurzelberechnungen nachträglich zu erhöhen. Das Heronverfahren besagt, dass die Folge

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right) \tag{1}$$

quadratisch gegen die Wurzel aus a konvergiert, d.h. mit jedem Schritt verdoppelt sich die Anzahl der signifikanten Stellen.

Das Verfahren lässt sich so verstehen: ist $x_n = \sqrt{a}$, so gilt $x_n = \frac{a}{x_n}$. Die neue Näherung wird daher als der Mittelwert aus diesen beiden Werten gebildet.

- **9.1.1** Implementiert das Heronverfahren in Python, um die Wurzel einer beliebigen Zahl auf 5 Nachkommastellen genau zu berechnen. (2 Punkte)

  Hinweise:

  - Auch der Computer kann natürlich nicht unendliche viele Folgenglieder von Gleichung (1) berechnen. Ihr müsst Euch also eine *Abbruchbedingung* überlegen, also, wann das Programm enden soll. Überlegt Euch, wann sich die neue Näherung nur noch marginal von der alten unterscheiden wird?
  - Stellt sicher, dass das Programm auch in der Lage ist, Fehler abzufangen, also wenn man sinnlose Werte als Eingabe gibt. Testet dazu das Programm mit den Werten $a \in \{0, 2, \pi, -1\}$.
- **9.1.2** Erweitert das Programm so, dass es die Wurzeln der Zahlen von 1 bis 10 berechnet. Nun schreibt ein Shellskript, das das Program ausführt und dann mit Hilfe von Gnuplot einen Plot dieser Daten erzeugt. (3 Punkte)

Aufgabe 9.2: Zahlensysteme (5 Punkte)

- **9.2.1** Berechnet die folgenden Zahlen a bis k, indem Ihr zwischen verschiedenen Zahlensystemen umrechnet. Dabei steht 1234_7 für die Zahl 1234 im Zahlensystem zur Basis 7. $a_{10} = 1234_7$ bedeutet also, daß die Zahl 1234 im Zahlensystem zur Basis 7 ins Zahlensystem zur Basis 10 (Dezimalsystem) umgerechnet werden soll. (2 Punkte)

  Hinweis: In einigen (aber nicht allen) Fällen dürfte die nebenstehende Tabelle nützlich sein.

  - $a_{10} = 1234_7$
  - $b_7 = 1234_{10}$
  - $c_{16} = 4321_7$
  - $d_{16} = 10000001_2$
  - $e_{16} = 10100101_2$
  - $f_8 = 10000001_2$
  - $g_8 = 10100101_2$
  - $h_2 = CD_{16}$
  - $i_2 = 27_8$

2	**7**	**8**	**10**	**16**
0	0	0	0	0
1	1	1	1	1
10	2	2	2	2
11	3	3	3	3
100	4	4	4	4
101	5	5	5	5
110	6	6	6	6
111	10	7	7	7
1000	11	10	8	8
1001	12	11	9	9
1010	13	12	10	A
1011	14	13	11	B
1100	15	14	12	C
1101	16	15	13	D
1110	20	16	14	E
1111	21	17	15	F
10000	22	20	16	10

- **9.2.2** Im Computerumfeld wird häufig das Hexadezimalsystem ($B = 16$) verwendet. Welchen Vorteil bietet das System gegenüber dem Dezimalsystem ($B = 10$) im Computerumfeld? Welchen Vorteil bietet es gegenüber dem Oktalsystem ($B = 8$)? (1 Punkt)

- **9.2.3** Schreibt ein Pythonprogramm, dass eine beliebige Zahl im Dezimalsystem ($B = 10$) in eine Zahl im Heptalsystem ($B = 7$) umwandelt. (2 Punkte)

  Hinweise:

  - Die „Eingabe“ des Programms soll einfach eine Variable sein:

    ```
    # Eingabe
    zahl_dezimal = 123456
    # Berechnung der einzelnen Stellen der Heptalzahl
    ```

  - Das Programm darf die Ziffern des Heptalsystems einzeln ausgeben, auch gerne in umgekehrter Reihenfolge (kleinste Stelle zuerst).