Abteilung für
Angewandte Mathematik

SoSe 2019
Christian Palus

Einführung in die Programmierung für Studierende der Naturwissenschaften

Blatt 3 – 13.05.2019

Abgabe: Briefkästen RZ/E-Mail bis Montag, den 20.05.2019, 16:00 Uhr

---

Aufgabe 1 (5 Punkte)**.** Ein *Flussdiagramm* (auch *Programmablaufplan*) dient der Übersichtlichen Darstellung von Algorithmen bzw. Programmabläufen. Es setzt sich aus den folgenden Elementen zusammen:

Zur Veranschaulichung finden Sie ein Flussdiagramm der Zählschleife aus dem Beispielprogramm `countdown-for.cc` auf der Rückseite des Übungsblatts. Zeichnen Sie ein Flussdiagramm des Programms mit dem Spiel zum Zahlenraten (vgl. Blatt 2, Aufgabe 4).

Aufgabe 2 (5 Punkte)**.** (i) Bestimmen Sie die Binärdarstellungen (d. h. die Darstellungen im Dualsystem) der Zahlen 7, 23, 109.
(ii) Erläutern Sie allgemein, wie man die Binärdarstellung einer positiven ganzen Zahl durch wiederholtes Dividieren mit Rest durch die Zahl 2 bestimmen kann.
(iii) Führen Sie die schriftliche Addition mit Übertrag von $11+25$ in Binärdarstellung durch.
(iv) Sei $0 \leq \ell \leq 2^k-1$ und $b = b_{k-1} \dots b_0$ die Binärdarstellung von ℓ. Die Binärzahl c entstehe aus b durch bitweises Invertieren und Addition von 1. Zeigen Sie, dass $b+c = 10\dots0$ mit k Nullen gilt. Durch Streichen des führenden Bits ergibt sich also Null.
(v) Führen Sie die Subtraktion zweier Binärzahlen am Beispiel $37-11$ mithilfe des Resultats aus (iv) auf eine geeignete Addition zurück.

Aufgabe 3 (5 Punkte)**.** Verändern Sie das Beispielprogramm `wurzel.cc` von der Vorlesungshomepage, indem Sie die Funktion `sqrt()` durch eine selbst geschriebene Funktion `double heron(double x, double tol)` ersetzen, welche das *Heron-Verfahren* zur näherungsweisen Berechnung von Quadratwurzeln realisiert. Implementieren Sie dazu in der selbst geschriebenen Funktion den folgenden Alorithmus:

Eingabe: Reelle Zahl $x \geq 0$ und Toleranz $\delta > 0$.

(1) Setze $s_{\text{neu}} = 1$ und $s_{\text{alt}} = x$.
(2) Gilt $|s_{\text{neu}} - s_{\text{alt}}| < \delta$, so stoppe.
(3) Setze $s_{\text{alt}} = s_{\text{neu}}$ und anschließend $s_{\text{neu}} = \frac{1}{2}(s_{\text{alt}} + \frac{x}{s_{\text{alt}}})$.
(4) Gehe zu Schritt (2).

Ausgabe: Approximation $s_{\text{neu}} \approx \sqrt{x}$.

Aufgabe 4 auf der Rückseite.

Aufgabe 4 (5 Punkte)**.** Schreiben Sie ein Programm, das ohne Verwendung der Potenzfunktion für eine einzugebende Zahl $n \geq 0$ die Summe

$$s(n) = \frac{4 \cdot (-1)^0}{2 \cdot 0 + 1} + \frac{4 \cdot (-1)^1}{2 \cdot 1 + 1} + \frac{4 \cdot (-1)^2}{2 \cdot 2 + 1} + \cdots + \frac{4 \cdot (-1)^n}{2 \cdot n + 1}$$

berechnet. Man kann nachweisen, dass diese Summen die Zahl

$$\pi = 3.1415926535\ 8979323846\ 2643383279\ 5028841971\ \ldots$$

beliebig gut approximieren. Testen Sie experimentell, welches n benötigt wird, um 6 Nachkommastellen exakt zu erhalten. Wie viele exakte Nachkommastellen erhalten Sie maximal bei Verwendung von Variablen des Typs `float` ? Wie viele sind es bei Verwendung des Typs `double` ?

Anhang zu Aufgabe 1. Flussdiagramm des Beispielprogramms `countdown-for.cc`: