Technische Universität München
Institut für Informatik
Prof. Dr. Hans-Joaching Bungartz
Dipl.-Tech. Math. Tobias Neckel
Dipl.-Tech. Math. Stefanie Schraufstetter

WS 2008/2009

Übungen zu Numerisches Programmieren

1. Programmieraufgabe

Gleitkommaarithmetik

Darstellung

Reelle Zahlen werden in der Informatik im Allgemeinen als Gleitpunktzahlen repräsentiert. Dabei wird jede Zahl $r \in \mathbb{R}$ durch ein Vorzeichen v, eine sogenannte Mantisse (lateinisch: *Zugabe*) m, einen Exponenten e und eine Basis $b \in \mathbb{N} \setminus \{1\}$ dargestellt:

$$x = (-1)^v \cdot m \cdot b^e \tag{1}$$

Im Folgenden beschränken wir uns auf die Darstellung im Zweiersystem; setzen also $b = 2$. Man beachte, dass durch die Definition (1) Mantisse und Exponent nicht eindeutig festgelegt sind. Z.B. gilt:

$$\cdots = 0.25 \cdot 2^6 = 0.5 \cdot 2^5 = 1 \cdot 2^4 = \cdots \tag{2}$$

Um eine eindeutige Darstellung zu erhalten, legen wir eine Normierung fest: Der Exponent e soll stets so gewählt werden, dass die Mantisse m genau eine von Null verschiedene Stelle vor dem Komma hat. Ausgenommen ist die Null, da bei ihr überhaupt keine Stelle der Mantisse von Null verschieden ist. Dies ist nur eine Möglichkeit, eine eindeutige Darstellung zu erhalten. Im IEEE-Standard 754 bzw. 854 wird eine andere Normierung verwendet (vgl. `http://grouper.ieee.org/groups/754/reading.html`).

Zusammenfassend kann also eine reelle Zahl r im Binärsystem dargestellt werden als

$$r = (-1)^v \cdot (r_{t-1}, r_{t-2} r_{t-3} \ldots r_1 r_0)_2 \cdot 2^e, \text{ mit } r_{t-1} \stackrel{!}{=} 1 \tag{3}$$

wobei $v \in \{0, 1\}$ das Vorzeichen $(-1)^v$, die Zahlen $r_i \in \{0, 1\}, i = 0, \ldots, t-1$ den Wert der Mantisse m und die Integer-Zahl e den Exponenten bestimmt. Für die Darstellung der Zahl 0 legen wir fest:

$$0 = (-1)^0 \cdot (0, 00 \ldots 00)_2 \cdot 2^0 \tag{4}$$

Realisierung von Operationen

Operationen auf Gleitkommazahlen werden in der Regel mit höherer Genauigkeit berechnet und das Ergebnis anschließend wieder auf eine Maschinenzahl gerundet. In der vorliegenden Aufgabe verwenden wir bei der Berechnung der Ergebnisse eine Mantisse mit doppelter Stellenzahl. So kann erst beim Runden des Ergebnisses ein Fehler entstehen. Konkret sollen nur die Addition und die Subtraktion reeller Zahlen implementiert werden. Sollen zwei reelle Zahlen m und n verrechnet werden, so fallen im wesentlichen folgende Schritte an:

1. Zunächst werden die Mantissen mit doppelter Genauigkeit gespeichert und anschließend wird eine der beiden Zahlen denormalisiert, so dass beide den selben Exponenten haben. Dabei wird wie folgt verfahren:
   - Sind beide Zahlen 0, so muss nicht denormalisiert werden.
   - Ist nur eine der beiden Zahlen 0, so wird diese denormalisiert.
   - Sind beide Zahlen ungleich 0, so wird die betragsmäßig kleinere Zahl denormalisiert.

   Dies geschieht in der Methode `denormalisiere`.
2. Die beiden Mantissen werden miteinander verrechnet. Dies geschieht in Abhängigkeit von der Operation in der Methode `add` bzw. `sub`.
3. Das Ergebnis wird nach Definition (3) bzw. (4) normalisiert. Dies geschieht in der Methode `normalisiere`.

Datenstrukturen

In der vorliegenden Programmieraufgabe soll eine Klasse `Reell` implementiert werden, die genau die Definitionen (1), (3) und (4) umsetzt. Demnach beinhaltet diese Klasse einen boolschen Wert für das Vorzeichen, ein Bitfeld für die Darstellung der Mantisse und einen Integerwert für den Exponenten. Für das Bitfeld wird eine eigene Klasse `BitFeld` implementiert, die entsprechende Basisoperationen bereithält.

Anzumerken ist, dass **vor** dem ersten Instanzieren eines Objektes der Klasse `Reell` die Möglichkeit besteht, die Anzahl der zu verwendenden Bits für die Darstellung der Mantisse anzugeben. Geschieht dies nicht, so wird standardmäßig mit 32 Bits gearbeitet. Nach der Wahl bzw. der ersten Instanzierung ist die Anzahl der Bits **nicht** mehr veränderbar (ansonsten müsste bei jeder Rechenoperation auf die unterschiedlichen Darstellungen der Operanden eingegangen werden). In der Folge kannn davon ausgegangen werden, dass innerhalb der Klasse `Reell` alle erzeugten Instanzen mit konstanter Anzahl von Bits `anzBitsMantisse` auftreten. Ausgenommen sind lediglich Instanzen, die für die Berechnung von Operationen verwendet werden. Diese arbeiten mit doppelter Stellenanzahl, das Ergebnis der Operationen wird beim Normalisieren aber wieder auf `anzBitsMantisse` Stellen gerundet.

Rundung

Da nicht jede reelle Zahl als Maschinenzahl dargestellt werden kann, muss man sich Gedanken darüber machen, wie solche Zahlen dargestellt werden können. Wir benötigen also eine Vorschrift, nach der nicht darstellbare Zahlen auf darstellbare Zahlen abgebildet werden. Dies geschieht durch Runden der Mantisse. Dabei wird anhand der ersten abgeschnittenen Stelle entschieden, ob auf- oder abgerundet wird. Es gilt:

$$rd(r) = (-1)^v \cdot 2^e \cdot \begin{cases} r_{t-1}, r_{t-2} \dots r_1 r_0 & \text{für } r_{-1} = 0 \text{ (abrunden)} \\ r_{t-1}, r_{t-2} \dots r_1 r_0 + 2^{-t+1} & \text{für } r_{-1} = 1 \text{ (aufrunden)} \end{cases} \quad (5)$$

Aufgaben

- Implementieren Sie in der Klasse `BitFeld` die Methoden `add` und `sub`.
- Testen Sie die Methoden mit selbstgewählten Beispielen. Verwenden Sie dazu die Methoden `setLong` und `toLong`.
- Implementieren Sie in der Klasse `Reell` die Methoden `denormalisiere` und `normalisiere`.
- Testen Sie eingehend Ihre Klasse `Reell` anhand selbstgewählter Beispiele.
- Testen Sie ausgiebig das Komplettprogramm mit verschiedenen Eingabeparametern.

Formalia

- Das Programmgerüst erhalten Sie auf der Webseite zur Vorlesung unter der Rubrik *Programmieraufgaben*.
- Ergänzen Sie das Programmgerüst bitte **nur an den dafür vorgegebenen Stellen!** Falls Sie die Struktur der Programme eigenmächtig verändern, können wir sie evtl. nicht mehr testen.
- **Beseitigen Sie vor Abgabe ihres Programms alle Ausgaben an die Konsole!**
- Eine Abgabe ist **ausschließlich** über das entsprechende Web-Formular möglich.
- Abgabeschluss ist **Montag, der 10. November 2008, 10:00 Uhr**.