Technische Universität München
Institut für Informatik
Prof. Dr. Thomas Huckle
Michael Lieb, M. Sc.
Dipl.-Tech. Math. Stefanie Schraufstetter

SS 2010

Numerisches Programmieren

1. Programmieraufgabe: Zahlendarstellung, Rundungsfehler

Gleitpunktarithmetik

Darstellung von Gleitkommazahlen

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

$$r = (-1)^v \cdot (r_{t-1}, r_{t-2}r_{t-3}\ldots r_1 r_0)_2 \cdot 2^{(e_{s-1}\ldots e_0)_2 - o}, \text{ mit } r_{t-1} \stackrel{!}{=} 1. \tag{3}$$

Hierbei bestimmt $v \in \{0,1\}$ das Vorzeichen $(-1)^v$, die Zahlen $r_i \in \{0,1\}, i = 0,\ldots,t-1$ den Wert der Mantisse m und $e_i \in \{0,1\}, i = 0,\ldots,s-1$ mit dem Offset o den Wert des Exponenten e.

Für die Darstellung der Zahl 0 legen wir fest:

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

Beschreibung des Programmgerüsts

Für die erste Programmieraufgabe wird Ihnen ein Programmgerüst zur Darstellung von Gleitpunktzahlen zur Verfügung gestellt. Es besteht aus den beiden Klassen *Gleitpunktzahl* und *BitFeld*. Wie man in Gleichung (3) sieht, werden die Mantisse und der Exponent durch t bzw. s Bits dargestellt.

Beachten Sie, dass der Exponent analog zum IEEE-Standard mit einem Offset versehen ist, um auch negative Exponenten darstellen zu können. Bei einem Exponenten mit acht Bit ist der kleinste darstellbare Exponent Null (alle acht Bit 0) und der größte Exponent ist 255 (alle acht Bit 1). Um den eigentlichen Exponenten zu bekommen, wird in diesem Fall 127 abgezogen (allgemein $2^{i-1} - 1$). Für die Zahl $1 = (-1)^0 \cdot 1 \cdot 2^0$ wird also der Exponent $(01111111)_2$ abgespeichert.

Die Klasse *BitFeld* dient zur Speicherung und Verarbeitung dieser Bitfelder. Es sind unter anderem bereits Methoden zum Addieren und Subtrahieren von Bitfeldern und zum Schieben von Bitfeldern nach links bzw. rechts implementiert.

Die Klasse *Gleitpunktzahl* dient zur Speicherung und Verarbeitung von Gleitpunktzahlen. Sie enthält Variablen vom Typ *BitFeld* für Mantisse und Exponent und eine Variable vom Typ *boolean* für das Vorzeichen. Es sind unter anderem Methoden zum Addieren, Subtrahieren, Normalisieren und Denormalisieren von Gleitpunktzahlen implementiert.

Vor dem Instanziieren des ersten Objekts der Klasse *Gleitpunktzahl* muss die Anzahl an Bits, die für die Speicherung von Mantisse und Exponent verwendet werden, festgelegt werden. Dazu werden die statischen Variablen *anzBitsMantisse* und *anzBitsExponent* mit den Methoden *setAnzBitsMantisse* und *setAnzBitsExponent* einmalig mit Werten belegt. Diese Werte dürfen nicht mehr geändert werden, damit sichergestellt ist, dass im Folgenden sämtliche Zahlen gleich aufgebaut sind.

Während der Berechnung von Operationen werden jedoch größere Mantissen benötigt, um sicherzustellen, dass eine ideale Gleitpunktarithmetik implementiert wird. Der Begriff **ideale Gleitpunktarithmetik** bedeutet, dass das berechnete Ergebnis genau dem gerundeten exakten Ergebnis entspricht. Nach dem Berechnen muss das Ergebnis wieder normalisiert werden.

Aufgabenstellung

Das gegebene Programmgerüst soll um die Multiplikation und Division von Gleitpunktzahlen ergänzt werden.

Erläuterungen

Bei der Addition von Gleitpunktzahlen wird zunächst die vom Betrag her kleinere Zahl denormalisiert, dann werden die beiden Mantissen addiert (bzw. subtrahiert falls eine der Zahlen negativ ist) und das Ergebnis wird wieder normalisiert. Bei der Multiplikation bzw. Division ist eine Denormalisierung nicht notwendig. Das Produkt der beiden Gleitpunktzahlen $x = (-1)^{v_1} \cdot m_1 \cdot 2^{e_1}$ und $y = (-1)^{v_2} \cdot m_2 \cdot 2^{e_2}$ lautet:

$$x \cdot y = (-1)^{v_1 \ XOR \ v_2} \cdot (m_1 \cdot m_2) \cdot 2^{e_1+e_2} \tag{5}$$

Analog gilt für die Division:

$$x \cdot y = (-1)^{v_1 \ XOR \ v_2} \cdot (m_1 \div m_2) \cdot 2^{e_1-e_2} \tag{6}$$

Sowohl bei der Multiplikation als auch bei der Division muss darauf geachtet werden, dass das berechnete Ergebnis normalisiert wird. Das normalisierte Ergebnis muss genau dem gerundeten exakten Ergebnis entsprechen (optimale Gleitpunktarithmetik).

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:

$$rd(r) = (-1)^v \cdot 2^e \cdot \begin{cases} r_{t-1}, r_{t-2} \ldots r_1 r_0 & \text{für } r_{-1} = 0 \text{ (abrunden)} \\ r_{t-1}, r_{t-2} \ldots r_1 r_0 + 2^{-t+1} & \text{für } r_{-1} = 1 \text{ (aufrunden)} \end{cases} \tag{7}$$

Sonderfälle

Es gibt drei Sonderfälle, die bei Bearbeitung der Aufgabe beachtet werden müssen:

- Unendlich: Der Maximale Wert den das BitFeld für den Exponent annehmen kann ist bei Verwendung von bspw. 8 Bit 255. Der Wert des eigentlichen

Exponenten ist somit $255 - \mathbf{o} = 255 - 127 = 128$. Alle größeren Exponenten können nicht mehr dargestellt werden. Da diese Zahlen dennoch mit ∞ repräsentiert werden sollten, verwenden wir den höchsten Exponenten für den speziellen Wert Unendlich. Die Mantisse wird komplett mit Nullen belegt. Das Vorzeichen legt fest, ob der Wert plus oder minus Unendlich ist. Sowohl bei der Multiplikation(zwei sehr große Zahlen) als auch bei der Division (Teilen durch Null) kann das Ergebnis den Wert Unendlich annehmen.

- NaN: Dieser Wert entsteht beim Teilen von null durch null. Er erhält ebenfalls den höchsten Exponenten und eine beliebige Mantisse ungleich null.
- Null: Eine normalisierte Gleitpunktzahl kann nicht den Wert null annehmen, da das höchstwertigste Bit der Mantisse eine 1 ist. Zur Darstellung der Null werden Mantisse und Exponent auf null gesetzt. (Siehe (4).)

Die von Ihnen implementierten Methoden müssen dazu in der Lage sein, auf beliebigen Gleitpunktzahlen inklusive der Null **(ohne die anderen Sonderfälle) als Eingabe** zu arbeiten und ein korrektes Ergebnis aus der Menge der Gleitpunktzahlen **inklusive aller genannten Sonderfälle** zu berechnen.

Aufgaben

- Programmieren Sie in der Klasse *BitFeld* die gegebenen Methodenrümpfe *mult* und *div* entsprechend der Beschreibungen in den Kommentaren.
- Programmieren Sie in der Klasse *Gleitpunktzahl* die Methodenrümpfe *mult* und *div* entsprechend der Beschreibungen in den Kommentaren.
- Testen Sie Ihre Implementierung anhand selbstgewählter Beispiele. Als Anregung für eine Testumgebung können Sie das Beispielprogramm *Aufgabe1.java* verwenden. Hierin sind bereits einige Testfälle für die Klasse Bitfeld implementiert.

Formalien und Hinweise

- Das Programmgerüst erhalten Sie auf den Webseiten zur Vorlesung.
- Ergänzen Sie das Programmgerüst bitte **nur an den dafür vorgegebenen Stellen!** Falls Sie die Struktur der Programme an anderer Stelle verändern, können wir sie evtl. nicht mehr testen.
- Beseitigen Sie vor Abgabe Ihres Programms alle Ausgaben an die Konsole und reichen sie bis zum **21. Mai 2010, 12:00 Uhr** über das Web-Portal (siehe Homepage) ein.
- Bei dieser Aufgabe kann es in Windows zu Problemen beim Testen von Rechenoperationen auf der Konsole kommen. Wir empfehlen daher, die Programme unter Linux (Rechnerhalle) zu testen.