Technische Universität München
Institut für Informatik
Prof. Dr. Hans-Joachim Bungartz
Dipl.-Inf. Christoph Riesinger
Dipl.-Math. Jürgen Bräckle

WS 2013/2014

Numerisches Programmieren, Übungen

1. Übungsblatt: Zahlendarstellung, Rundungsfehler

1) Umrechnung von Zahlen

a) Schreiben Sie die ganzen Zahlen 19, 47 und 511 in binärer, trinärer und hexadezimaler Darstellung!

b) Schreiben Sie die Brüche $-\frac{1}{7}$ und $\frac{1}{10}$ als Binärzahl mit Nachkommastellen!
(Bsp: $-\frac{1}{4} = -0.01_2$)

2) Binärdarstellung von ganzen Zahlen

Ganze Zahlen werden im Computer in binärer Form gespeichert. Stehen n Bits zur Verfügung, so haben sie bei 2-Komplement-Darstellung folgende Wertigkeiten:

Stelle des Bits	1	2	3	...	n
Wert des Bits	-2^{n-1}	2^{n-2}	2^{n-3}	...	2^0

Bei einer Codierung mit 4 **Bits** ergäben sich zum Beispiel die folgenden Darstellungen:

Dezimalzahl	0	2	4	7	-7
Bitcodierung	0000	0010	0100	0111	1001

Geben Sie die untere und obere Grenze für den Zahlbereich an, der mit Hilfe von n **Bits** dargestellt werden kann! Wie sieht der Zahlenbereich speziell für 4 Byte-Integer aus?

3) Gleitkomma-Zahlen

In dieser Aufgabe soll Schritt für Schritt eine reelle Zahl in eine Maschinenzahl umgewandelt werden. Die umzuwandelnde Zahl lautet $-\frac{11}{10}$.

a) Schreiben Sie die Zahlen zuerst in eine andere, standardisierte Form folgender Gestalt um! An erster Stelle steht das Vorzeichen. Dann folgt der Betrag der Zahl in binärer Exponentialdarstellung, beginnend mit "$1,\ldots$". Es geht weiter mit den Nachkommastellen, einem Malpunkt und abschließend steht eine Zweierpotenz (Beispiel: $-1{,}11001 \cdot 2^{-56}$).

b) Wandeln Sie die Ergebnisse von Teilaufgabe a) in Binärcode um! Dazu verwenden Sie den im Folgenden spezifizierten 32-Bit IEEE-Standard:

- Das erste Bit bestimmt das Vorzeichen: Eine Null bedeutet positive Zahl, eine Eins bedeutet negative Zahl.
- Die nächsten 8 Bits sind für den Exponenten reserviert. Die gespeicherte Binärzahl entspricht dem Exponenten plus 127. Die Bit-Kombinationen 00000000 (entspräche einem Exponenten von -127) und 11111111 (entspräche $+128$) sind allerdings für spezielle Werte reserviert (0, Inf, NaN).
- Die letzten 23 Bits dienen der Speicherung der Nachkommastellen. Dabei wird wie in Teilaufgabe a) von einer Normalisierung mit führender Eins ausgegangen (die nicht gespeichert werden muss). Genügen 23 Bits Genauigkeit für die Mantisse nicht, so wird zum nächstgelegenen darstellbaren Wert gerundet. Beispiele:
  - $x = 1,0|\mathbf{101} \rightarrow$ Aufrunden (1.1)
  - $x = 1,0|\mathbf{0}11 \rightarrow$ Abrunden (1.0)
  - Uneindeutiger Fall $|x_t x_{t+1} x_{t+2} \ldots = 100\ldots$
    - $x = 1,\mathbf{1}|100 \rightarrow$ Aufrunden (10.0)
    - $x = 1,\mathbf{0}|100 \rightarrow$ Abrunden (1.0)
- Es gäbe noch viel über IEEE zu sagen, was aber nicht zur Lösung dieser Aufgabe notwendig ist. Weitere Informationen finden Interessierte zum Beispiel im Artikel *What Every Computer Scientist Should Know about Floating-Point Arithmetic* von David Goldberg (siehe z.B. `http://grouper.ieee.org/groups/754/`).

c) Geben Sie eine Zahl zwischen 1 und 2 an, die mit dem in Teilaufgabe b) beschriebenem IEEE-Standard nicht exakt darstellbar ist! Schätzen Sie zudem den absoluten und relativen Rundungsfehler ab!

d) Wie groß ist die Maschinengenauigkeit ε_{Ma} für den in Teilaufgabe b) beschriebenen IEEE-Standard?

4) Fehlerabschätzung von Zeitschrittweiten

In Computerspielen werden für die Simulation von Physik oft Zeitschrittweiten Δt von $\frac{1}{50}$ oder $\frac{1}{60}$ gewählt um die Positionsänderung von einem Objekt innerhalb eines Frames zu simulieren.

Wir wollen hier den vereinfachten Fall mit konstanter Objekt-Geschwindigkeit und unter Vernachlässigung aller externen Kräfte und Kollisionen betrachten. Dann lässt sich die Positionsänderung eines Objektes innerhalb eines Zeitschrittes durch $\Delta x = \Delta t \cdot \vec{v}$ beschreiben, wobei Δx die Positionsänderung innerhalb eines Zeitschrittes ist und $\vec{v}$ die konstante Geschwindigkeit des Objektes.

Durch die vorangegangenen Aufgaben haben wir bereits festgestellt, dass Zahlen in Binärdarstellung nur eine begrenzte Genauigkeit besitzen, womit Δt evtl. nicht exakt gespeichert werden kann.

Geben Sie eine Abschätzung an, ob und wieviel Genauigkeit bei der Konvertierung der Zeitschrittweite $\frac{1}{60}$ zu 32-bit IEEE floating-point Zahlen verloren geht.

Inwiefern kann sich dieser Fehler auf sehr lange Simulationsläufe auswirken? Haben Sie eine Idee, dieses Problem zu umgehen bzw. den entstehenden Fehler zu klein zu halten?

5) Ermittlung von π nach Archimedes

Viele mathematische und naturwissenschaftliche Probleme können nicht oder nur schwer durch Angabe einer direkten Lösungsformel gelöst werden, stattdessen ist aber oftmals eine schrittweise Annäherung an die exakte Lösung möglich. Solche sogenannten iterativen Approximationen lassen sich meist algorithmisch beschreiben und in ein Computer-Programm umsetzen. Dabei ist allerdings große Sorgfalt geboten, wie die folgende Aufgabe zeigt.

Ein klassisches Beispiel für die iterative Approximation ist die Bestimmung der Kreiszahl π. Eines der ersten bekannten Verfahren geht auf Archimedes von Syrakus (um 250 v.Chr.) zurück. Die Formel Kreisumfang gleich zweimal Kreisradius mal π war Archimedes bereits bekannt. Durch immer genauere Approximation des Umfangs eines Kreises mit Radius eins mit Hilfe von in den Kreis einbeschriebenen Polygonen konnte er somit π approximieren. Für ihn war das eine mühselige Arbeit mit Tinte und Papyrus, wir können das heute dank Computer im Bruchteil einer Sekunde erledigen.

a) Berechnen Sie Seitenlänge und Umfang eines in den Einheitskreis einbeschriebenen Quadrats (vgl. Abbildung 1)!

b) Durch Verdopplung der Ecken erhält man aus dem Quadrat sukzessive regelmäßige Achtecke, Sechszehnecke, etc. (vgl. Abbildung 1). Geben Sie eine Formel für die Seitenlänge s_{n+1} des 2^{n+1}-Ecks in Abhängigkeit von der Seitenlänge s_n des 2^n-Ecks an!
Hinweis: Fertigen Sie eine Skizze an und erinnern Sie sich an den Satz von Pythagoras!

c) Mit der in Teilaufgabe b) gefundenen iterativen Formel für die Seitenlänge der Polygone, kann man nun schrittweise Näherungen für π berechnen. Implementiert man die Formel zum Beispiel in einer `FOR`-Schleife, so ergeben sich aber folgende Werte:

Anzahl der Ecken	2^5	2^{10}	2^{20}	2^{25}	2^{30}
Approximation	$3,136548$	$3,141587$	$3,141596$	$3,142451$	$0,000000$

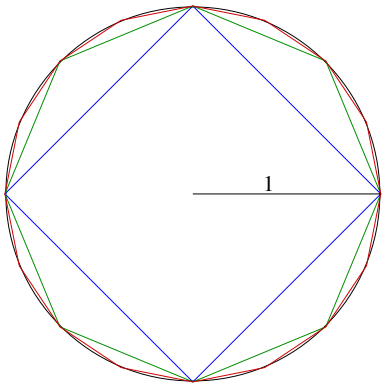

Abbildung 1: Einheitskreis und einbeschriebene Polygone

Welcher Teil der Formel aus Teilaufgabe b) ist für den auftretenden Fehler verantwortlich? Beheben Sie das Problem durch algebraische Umformungen!