Numerisches Programmieren

Übung #01

geheimtippmuenchen.de/content/uploads/2023/02/geheimtipp-muenchen-thiersch-turm-1.jpg

Organisatorisches

- Übungen
  - **2 Stunden** lang
  - Übungswochen von **Mi. bis Di.**
- 4 Programmier-Hausaufgaben
  - 3er Gruppen, Anmeldung über Moodle
  - mind. **70%** der Gesamtpunkte (280/400) für 0.3 Notenbonus
- Folien
  - Auf Moodle sichtbar
  - Viele Grafiken aus dem Hendrik Möller Skript

About me

👨🏻 Sam Miao

🎓 B.Sc. Informatik, 8. Semester

✉️ sam.miao@tum.de

Zahlendarstellung

Zahlensysteme

- Basis B (≅ Anzahl Ziffern)
- Berechnung einer Zahl:

$$x = \sum_{i=0}^{n-1} x_i \cdot B^i$$

  - x_i ist Ziffer ∈ $[0, B-1]$, B^i ist Stellenwertigkeit an Position i
- Beispiel:

$x = 384;\ B = 10$: $384 = 3 \cdot 10^2 + 8 \cdot 10^1 + 4 \cdot 10^0$

Wichtige Zahlenbasen

- **Dezimalzahlen**: Basis 10
- **Binärzahlen**: Basis 2
  - Ziffern 0 und 1
  - Beispiel: 42 = 0b101010
- **Hexadezimalzahlen**: Basis 16
  - Ziffern mit Dezimalwert > 9 als Buchstaben codiert
  - Beispiel: 42 = 0×2A

Hex-Buchstabe	A	B	C	D	E	F
Dezimalwert	10	11	12	13	14	15

Tabelle 3: Wdh. Hex Buchstaben.

Dezimal ➡ Binär

Algorithmus:

1. Zahl durch 2 teilen
2. Rest notieren
3. Zurück zu Schritt 1. bis die Zahl 0 ist

- Die Ziffern vom Ergebnis sind die Reste von *unten nach oben*

Beispiel: $x = 43$

$43/2 = 21$, Rest 1

$21/2 = 10$, Rest 1

$10/2 = 5$, Rest 0

$5/2 = 2$, Rest 1

$2/2 = 1$, Rest 0

$1/2 = 0$, Rest 1

→ Ergebnis: $0b101011$

Binär ↔ Hex

Binär → Hex:

1. Binärzahl aufteilen in „4er-Blöcke“ (rechts nach links)
2. Binärblöcke in Hex-Ziffern umwandeln (eindeutiges Mapping mögl.)

- Beispiel:

0b1101001101 → 0011 0100 1101 → 3 4 D → 0×34D

Hex → Binär: Das Ganze rückwärts

Nachkommastellen in Binär

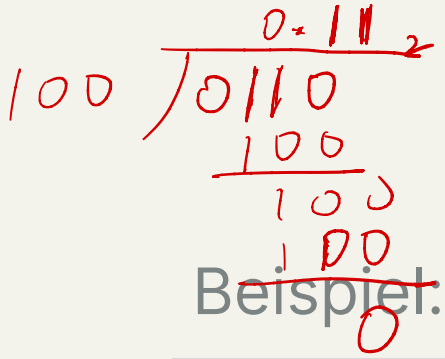
- Zweierpotenzen mit negativen Exponenten (0.5, 0.25, 0.125, ...) kommen dazu
- Aus welchen dieser Zweierpotenzen kann ich die Nachkommastellen „zusammensetzen“?
- Beispiel:

$$0.625 = 0.5 + 0.125 = 2^{-1} + 2^{-3} = 0.101_2$$

Binär	1.0_2	0.1_2	0.01_2	0.001_2	0.0001_2	0.00001_2	...
Dezimalwert	1	$1/2$	$1/4$	$1/8$	$1/16$	$1/32$	...

Tabelle 5: Wdh. Binäre Kommazahlen

Brüche in Binär



Algorithmus:

1. Zähler & Nenner jeweils in Binärzahl umwandeln
2. Schriftliches Dividieren

- Das Minus wird als Minus gelassen

Beispiel: $x = 3/4$

$\frac{3}{4} = 011_2 : 100_2$

0	1	1	|		$: 100_2 = 0.11_2$
	1	1	0		
-	1	0	0		
		1	0	0	
	-	1	0	0	
				0	

→ Ergebnis: 0.11_2

Aufgabe 1)

1) Umrechnung von Zahlen

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

b) Schreiben Sie die Brüche $-\frac{3}{8}$ und $\frac{1}{10}$ als Binärzahl!

Runden von Zahlen

- Speicherung von Zahlen im Computer
  - Als Binärzahlen
  - Nur begrenzt viele Bits pro Zahl zur Verfügung

→ Zahlen müssen gerundet werden: $\boldsymbol{rd(x)}$

→ Information geht dadurch verloren

- **Absoluter Fehler:** $f_{abs} := |x - rd(x)|$
- **Relativer Fehler:** $f_{rel} := |\frac{f_{abs}}{x}|$

Binärzahlen runden

Vier Fälle:

Sei y beliebig und Zahl bis zum „|“ korrekt darstellbar

a. $y|0x,$ x beliebig → Abrunden

b. $y|1x,$ x nicht nur Nullen → Aufrunden

c. $y1|1x,$ x nur Nullen → Aufrunden

d. $y0|1x,$ x nur Nullen → Abrunden

→ c. und d. immer zur geraden Mantisse gerundet!

Aufgabe 2)

2) Assoziativgesetz

Eine Eigenschaft, die bei Gleitkommazahlen leider verloren wird, ist das Assoziativgesetz. Betrachten Sie eine binäre Gleitkomma-Darstellung mit 4 signifikanten Stellen.

	Zahl	Anzahl signifikanter Stellen
Beispiele:	1000000.0_2	8
	$1 \cdot 2^6$	1
	1000.1_2	5
	-0.01011_2	4
	10.10_2	4

Berechnen Sie im gegebenen Format die Werte

- $(-8 + 11) + 0.75$ und
- $-8 + (11 + 0.75)$.

Runden Sie dabei nach jedem Rechenschritt. Was ist zu beobachten?

Absorption

カハ不変

Definition: *Addition zweier (sehr) unterschiedlich großer Zahlen*

- Beispiel (nur 3 Ziffern speicherbar):

$$rd(11_2 + 0.01_2) = rd(11.01_2) = 11_2$$

- Information geht verloren weil gerundet werden muss
  - „Absorption“ der kleineren Zahl
  - Manchmal lösbar durch Änderung der Additionsreihenfolge

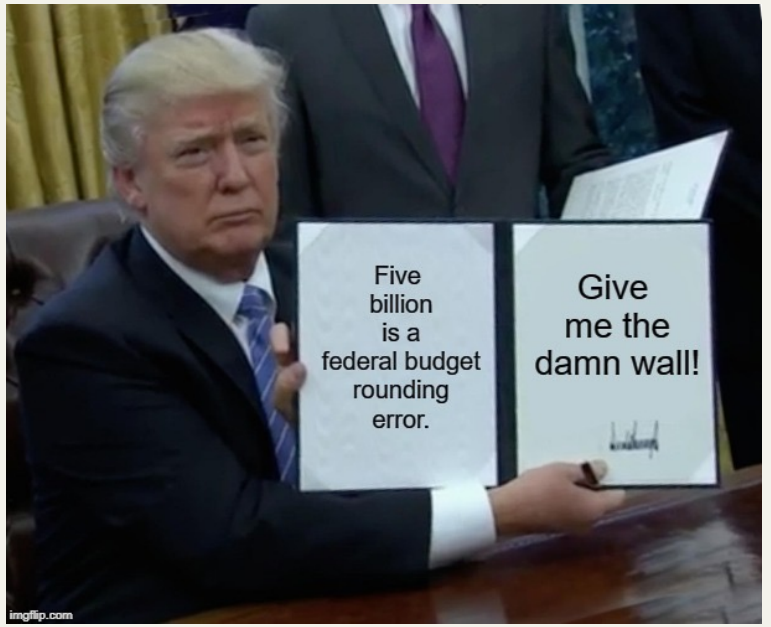
Auslöschung

减为0

Definition: *Subtraktion zweier fast gleich großer Zahlen*

Beispiel (nur 2 Ziffern speicherbar):

$$rd(6.942) - rd(6.901) = 6.9 - 6.9 = 0$$



https://i.imgflip.com/2plr82.jpg

- Kann sich zu einer Null runden
- Potenziell gefährlicher als Absorption

Zweierkomplement

- Möglichkeit auch negative Zahlen zu speichern
- Zahl negativ → Vorderstes Bit eine 1
- Vorderstes Bit repräsentiert -2^{n-1}

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

Beispiele (insg. 4 Bits):

- $0b1001 = -7$
- $0b0101 = 5$
- $0b1111 = -1$

Negation einer Zahl:

1. Bits invertieren
2. Eins aufaddieren

- Beispiel (insg. 4 Bits):
  0111_2 (7) → 1000_2 → 1001_2 (-7)

Gleitkommazahlen

- Man möchte beliebige reelle Zahlen darstellen können
  - Festkommazahlen können nur begrenzt viele Nachkommastellen speichern

Gleitkommazahlen:

- 3 Komponenten: Vorzeichen, Exponent, Mantisse
- Format: $G = (-1)^{V} \cdot 2^{E} \cdot M$ → „Zweierpotenz mal irgendeine Zahl“
- Beispiel:

$$-20 = -16 \cdot 1.25 = (-1)^{1} \cdot 2^{4} \cdot 1.01_{2}$$

- Normalisierung für Eindeutigkeit → 1 vorm Komma der Mantisse!

IEEE-Standard

- 32 Bits zur Verfügung
- Speicherlayout:
  - Vorzeichen **1** Bit, Exponent **8** Bits, Mantisse **23** Bits

Exponent:

- Soll negative Werte annehmen können (für Zahlen nahe bei 0)
- Subtraktion um 127 (der **Bias**), um den tatsächlichen Exponenten zu erhalten
- Spezielle Bitfolgen sind reserviert (z.B. 00000000, 11111111)

IEEE-Standard

Mantisse

- Vorderste 1 der Mantisse nicht gespeichert → 1 Bit mehr verfügbar
- Rundung der Mantisse, falls Bits nicht ausreichen

Dezimal	Binär	Gleitkommadarstellung	IEEE (V | E | M)
32.00	100000_2	$2^5 \cdot 1.0_2$	0|10000100|00000000000000000000000
13.75	1101.11_2	$2^3 \cdot 1.10111_2$	0|10000010|10111000000000000000000
-0.125	-0.001_2	$-1 \cdot 2^{-3} \cdot 1.0_2$	1|01111100|00000000000000000000000

Tabelle 8: Beispiele von Dezimalzahlen und ihre dazugehörige normierte Gleitkommadarstellung und wie die 32 Bits in IEEE aussehen würden.

Maschinengenauigkeit

t 23.

$\underline{Definition}$: $größte\ Zahl\ x, sodass\ rd(x+1) = 1\ (abgerundet)$

- Maß für Rundungsfehler
  - Maschine ist „genauer“ je kleiner x ist
- Nächstgrößere Zahl führt zu Aufrundung
- Berechnung durch $\epsilon = \frac{1}{2} \cdot B^{1-t}$, $t \coloneqq Mantissenlänge$
- Ein Algorithmus kann nicht genauer sein als Maschinengenauigkeit

Aufgabe 3)

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:

Aufgabe 3)

c) Wir haben nun eine Zahl gesehen, welche mit dem in Teilaufgabe b) beschriebenem IEEE-Standard nicht exakt darstellbar ist. Wir wollen nun den Fehler einer solchen Rundung berechnen. Zur Vereinfachung der Rechnung werden wir uns die Zahl $x = 1 + 2^{-30}$ anschauen. Wie sieht die gerundete Zahl aus? Berechnen sie außerdem den absoluten und relativen Fehler.

d) Wie groß ist die Maschinengenauigkeit ε_{Ma} für den in Teilaufgabe b) beschriebenen IEEE-Standard? Durch welche Größe wird die Maschinengenauigkeit verändert? Welche andere Größe gibt es und was wird hierdurch reguliert?

e) Wie unterscheidet sich der Zahlenbereich eines IEEE **float** und eines **signed int**? Geben Sie hierfür den Zahlenbereich der Gleitkommazahlen getrennt für positive und negative Zahlen an! Geben Sie außerdem an wie sich die Schrittweite der Diskretisierung verhält.

f) Geben sie die erste Ganzzahl an die mithilfe des 32-Bit IEEE-Standards für Gleitkommazahlen nichtmehr exakt darstellbar ist. Welche Probleme entstehen hieraus für Programmiersprachen, welche nur mit Gleitkommazahlen arbeiten und keine Integerzahlen besitzen (z.B. Javascript).

Danke fürs Kommen!
Bis nächste Woche! 😊

geheimtippmuenchen.de/content/uploads/2023/02/geheimtipp-muenchen-thiersch-turm-1.jpg