

Übungsstunde — Informatik — 06

Adel Gavranović

Follow-up, Referenzen, `std::vector`, Mehrdimensionale Vektoren

Übersicht

Follow-up

Prüfungsfrage

Runden (in F^*)

Referenzen

`std::vector<T>`

Multidimensionale Vektoren

Repetition: Fließkommazahlen

Alte Prüfungsfragen



n.ethz.ch/~agavranovic

[Material](#)

[Webpage](#)

[Mail](#)

1. Follow-up

Follow-up aus letzter Übungsstunde

Follow-up aus letzter Übungsstunde

- Entschuldigt die suboptimale Übungsstunde von letzter Woche

Follow-up aus letzter Übungsstunde

- Entschuldigt die suboptimale Übungsstunde von letzter Woche
- Sehr viel Follow-up heute

1. Follow-up

1.1. Prüfungsfrage

Prüfungsrelevant?

- Das ist eine echte Prüfungsaufgabe aus dem Jahr 2022
- Öffnet die Aufgabe "[Exam 2022.02 (MAVT + ITET)] Decimal to arbitrary base" auf **code expert**
- Besprecht die Aufgabe und euren Lösungsansatz mit euren Nachbar:innen

Prüfungsrelevant?

- Das ist eine echte Prüfungsaufgabe aus dem Jahr 2022
- Öffnet die Aufgabe "[Exam 2022.02 (MAVT + ITET)] Decimal to arbitrary base" auf **code expert**
- Bespricht die Aufgabe und euren Lösungsansatz mit euren Nachbar:innen
- Löst die Aufgabe

Lösung

Lösung

```
int convert_base(int n, int b) {
```

```
    int result = 0;
```

// result stored in base 10

```
    int basetenposition = 1;
```

// for setting correct digit in 'result'

```
    while (n != 0) {
```

```
        int rightmost_digit = (n % b);
```

```
        result += basetenposition * rightmost_digit;
```

```
        n = n / b;
```

← shift to left by 10

```
        basetenposition *= 10;
```

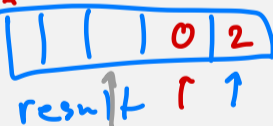
```
    }
```

```
    return result;
```

```
}
```

PRE n:
b: base 10

131.3
10 3
2 1.10



131.3 → 131

Fragen/Unklarheiten?

1. Follow-up

1.2. Runden (in F^*)

Wie Runden?

¹  754-2019 - IEEE Standard for Floating-Point Arithmetic

Wie Runden?

Fließkommazahlen runden ist *sehr kompliziert*¹

¹  754-2019 - IEEE Standard for Floating-Point Arithmetic

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

1. Abrunden falls "Cutoff-Zahl"

$$c < 1$$

c 1.00101...

1.1100101...

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

1. Abrunden falls "Cuttoff-Zahl"

$$c < 1$$

2. Aufrunden falls $c > 1$

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

1. Abrunden falls "Cutoff-Zahl"
 $c < 1$
2. Aufrunden falls $c > 1$
3. Falls $c = 1.000\dots$ dann "Round to even", i.e. einfach so runden, dass die letzte ziffer gerade ist (binär also immer 0 – aber vorsichtig!)

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

1. Abrunden falls "Cutoff-Zahl"
 $c < 1$
2. Aufrunden falls $c > 1$
3. Falls $c = 1.000\dots$ dann "Round to even", i.e. einfach so runden, dass die **letzte ziffer** gerade ist (binär also immer 0 – aber vorsichtig!)

■ Seien die exakten Resultate:

1. $1.01011 \cdot 2^2$
2. $1.01010 \cdot 2^2$
3. $1.01110 \cdot 2^2$
4. $1.01100 \cdot 2^2$

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

1. Abrunden falls "Cutoff-Zahl"
 $c < 1$
2. Aufrunden falls $c > 1$
3. Falls $c = 1.000\dots$ dann "Round to even", i.e. einfach so runden, dass die letzte ziffer gerade ist (binär also immer 0 – aber vorsichtig!)

■ Seien die exakten Resultate:

1. $1.01011 \cdot 2^2$
2. $1.01010 \cdot 2^2$
3. $1.01110 \cdot 2^2$
4. $1.01100 \cdot 2^2$

■ ...aber das sind mehr als $p = 4$ Ziffern!² Wir runden – aber wie?

1. $1.01011 \cdot 2^2$
↓
1.1

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

1. Abrunden falls "Cutoff-Zahl"
 $c < 1$
2. Aufrunden falls $c > 1$
3. Falls $c = 1.000\dots$ dann "Round to even", i.e. einfach so runden, dass die letzte Ziffer gerade ist (binär also immer 0 – aber vorsichtig!)

■ Seien die exakten Resultate:

1. $1.01011 \cdot 2^2$
2. $1.01010 \cdot 2^2$
3. $1.01110 \cdot 2^2$
4. $1.01100 \cdot 2^2$

■ ...aber das sind mehr als $p = 4$ Ziffern!² Wir runden – aber wie?

1. $1.01011 \cdot 2^2 \approx 1.011$ $p=4$
 $p=4$
 456.47

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

1. Abrunden falls "Cutoff-Zahl"

$$c < 1$$

2. Aufrunden falls $c > 1$

3. Falls $c = 1.000\dots$ dann "Round to even", i.e. einfach so runden, dass die letzte Ziffer gerade ist (binär also immer 0 – aber vorsichtig!)

- Seien die exakten Resultate:

1. $1.01011 \cdot 2^2$

2. $1.01010 \cdot 2^2$

3. $1.01110 \cdot 2^2$

4. $1.01100 \cdot 2^2$

- ...aber das sind mehr als $p = 4$ Ziffern!² Wir runden – aber wie?

1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

1. Abrunden falls "Cuttoff-Zahl"
 $c < 1$
2. Aufrunden falls $c > 1$
3. Falls $c = 1.000\dots$ dann "Round to even", i.e. einfach so runden, dass die **letzte ziffer** gerade ist (binär also immer 0 – aber vorsichtig!)

■ Seien die exakten Resultate:

1. $1.01011 \cdot 2^2$
2. $1.01010 \cdot 2^2$
3. $1.01110 \cdot 2^2$
4. $1.01100 \cdot 2^2$

■ ...aber das sind mehr als $p = 4$ Ziffern!² Wir runden – aber wie?

1. $1.010\underline{11} \cdot 2^2 \approx 1.011$ per (2)
2. $1.010\underline{10} \cdot 2^2$

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

1. Abrunden falls "Cutoff-Zahl"
 $c < 1$
2. Aufrunden falls $c > 1$
3. Falls $c = 1.000\dots$ dann "Round to even", i.e. einfach so runden, dass die letzte Ziffer gerade ist (binär also immer 0 – aber vorsichtig!)

■ Seien die exakten Resultate:

1. $1.01011 \cdot 2^2$
2. $1.01010 \cdot 2^2$
3. $1.01110 \cdot 2^2$
4. $1.01100 \cdot 2^2$

■ ...aber das sind mehr als $p = 4$ Ziffern!² Wir runden – aber wie?

1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)
2. $1.01010 \cdot 2^2 \approx 1.010$

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

1. Abrunden falls "Cuttoff-Zahl"
 $c < 1$
2. Aufrunden falls $c > 1$
3. Falls $c = 1.000\dots$ dann "Round to even", i.e. einfach so runden, dass die letzte Ziffer gerade ist (binär also immer 0 – aber vorsichtig!)

■ Seien die exakten Resultate:

1. $1.01011 \cdot 2^2$
2. $1.01010 \cdot 2^2$
3. $1.01110 \cdot 2^2$
4. $1.01100 \cdot 2^2$

■ ...aber das sind mehr als $p = 4$ Ziffern!² Wir runden – aber wie?

1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)
2. $1.01010 \cdot 2^2 \approx 1.010$ per (3)

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

1. Abrunden falls "Cuttoff-Zahl"
 $c < 1$
2. Aufrunden falls $c > 1$
3. Falls $c = 1.000\dots$ dann "Round to even", i.e. einfach so runden, dass die letzte Ziffer gerade ist (binär also immer 0 – aber vorsichtig!)

■ Seien die exakten Resultate:

1. $1.01011 \cdot 2^2$
2. $1.01010 \cdot 2^2$
3. $1.01110 \cdot 2^2$
4. $1.01100 \cdot 2^2$

■ ...aber das sind mehr als $p = 4$ Ziffern!² Wir runden – aber wie?

1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)
2. $1.01010 \cdot 2^2 \approx 1.010$ per (3)
3. $1.01110 \cdot 2^2$

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

1. Abrunden falls "Cuttoff-Zahl"
 $c < 1$
2. Aufrunden falls $c > 1$
3. Falls $c = 1.000\dots$ dann "Round to even", i.e. einfach so runden, dass die **letzte ziffer** gerade ist (binär also immer 0 – aber vorsichtig!)

■ Seien die exakten Resultate:

1. $1.01011 \cdot 2^2$
2. $1.01010 \cdot 2^2$
3. $1.01110 \cdot 2^2$
4. $1.01100 \cdot 2^2$

■ ...aber das sind mehr als $p = 4$ Ziffern!² Wir runden – aber wie?

1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)
2. $1.01010 \cdot 2^2 \approx 1.010$ per (3)
3. $1.01110 \cdot 2^2 \approx 1.\underline{100}$

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

1. Abrunden falls "Cutoff-Zahl"
 $c < 1$
2. Aufrunden falls $c > 1$
3. Falls $c = 1.000\dots$ dann "Round to even", i.e. einfach so runden, dass die **letzte ziffer** gerade ist (binär also immer 0 – aber vorsichtig!)

■ Seien die exakten Resultate:

1. $1.01011 \cdot 2^2$
2. $1.01010 \cdot 2^2$
3. $1.01110 \cdot 2^2$
4. $1.01100 \cdot 2^2$

■ ...aber das sind mehr als $p = 4$ Ziffern!² Wir runden – aber wie?

1. $1.010\underline{11} \cdot 2^2 \approx 1.011$ per (2)
2. $1.010\underline{10} \cdot 2^2 \approx 1.010$ per (3)
3. $1.011\underline{10} \cdot 2^2 \approx 1.\underline{100}$ per (3)
 $p=4$

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

1. Abrunden falls "Cutoff-Zahl"
 $c < 1$
2. Aufrunden falls $c > 1$
3. Falls $c = 1.000\dots$ dann "Round to even", i.e. einfach so runden, dass die letzte Ziffer gerade ist (binär also immer 0 – aber vorsichtig!)

■ Seien die exakten Resultate:

1. $1.01011 \cdot 2^2$
2. $1.01010 \cdot 2^2$
3. $1.01110 \cdot 2^2$
4. $1.01100 \cdot 2^2$

■ ...aber das sind mehr als $p = 4$ Ziffern!² Wir runden – aber wie?

1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)
2. $1.01010 \cdot 2^2 \approx 1.010$ per (3)
3. $1.01110 \cdot 2^2 \approx 1.100$ per (3)
4. $1.01100 \cdot 2^2$

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

1. Abrunden falls "Cuttoff-Zahl"
 $c < 1$
2. Aufrunden falls $c > 1$
3. Falls $c = 1.000\dots$ dann "Round to even", i.e. einfach so runden, dass die letzte ziffer gerade ist (binär also immer 0 – aber vorsichtig!)

■ Seien die exakten Resultate:

1. $1.01011 \cdot 2^2$
2. $1.01010 \cdot 2^2$
3. $1.01110 \cdot 2^2$
4. $1.01100 \cdot 2^2$

■ ...aber das sind mehr als $p = 4$ Ziffern!² Wir runden – aber wie?

1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)
2. $1.01010 \cdot 2^2 \approx 1.010$ per (3)
3. $1.01110 \cdot 2^2 \approx 1.100$ per (3)
4. $1.01100 \cdot 2^2 \approx 1.011$

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Wie Runden?

Solange nichts anderes von euch (in der Aufgabenbeschreibung) gefordert wird, könnt so runden wie in der Mathematik:

1. Abrunden falls "Cuttoff-Zahl"
 $c < 1$
2. Aufrunden falls $c > 1$
3. Falls $c = 1.000\dots$ dann "Round to even", i.e. einfach so runden, dass die letzte ziffer gerade ist (binär also immer 0 – aber vorsichtig!)

■ Seien die exakten Resultate:

1. $1.01011 \cdot 2^2$
2. $1.01010 \cdot 2^2$
3. $1.01110 \cdot 2^2$
4. $1.01100 \cdot 2^2$

■ ...aber das sind mehr als $p = 4$ Ziffern!² Wir runden – aber wie?

1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)
2. $1.01010 \cdot 2^2 \approx 1.010$ per (3)
3. $1.01110 \cdot 2^2 \approx 1.100$ per (3)
4. $1.01100 \cdot 2^2 \approx 1.011$ per (1)

²Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Beispiele

³Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Beispiele

- Seien die exakten Resultate der Berechnungen

1. $1.01011 \cdot 2^2$

2. $1.01010 \cdot 2^2$

3. $1.01110 \cdot 2^2$

4. $1.01100 \cdot 2^2$

³Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Beispiele

- Seien die exakten Resultate der Berechnungen
 1. $1.01011 \cdot 2^2$
 2. $1.01010 \cdot 2^2$
 3. $1.01110 \cdot 2^2$
 4. $1.01100 \cdot 2^2$
- ...aber das sind mehr als $p = 4$ Ziffern!³ Wir runden – aber wie?
 1. $1.01011 \cdot 2^2$

³Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Beispiele

- Seien die exakten Resultate der Berechnungen
 1. $1.01011 \cdot 2^2$
 2. $1.01010 \cdot 2^2$
 3. $1.01110 \cdot 2^2$
 4. $1.01100 \cdot 2^2$
- ...aber das sind mehr als $p = 4$ Ziffern!³ Wir runden – aber wie?
 1. $1.01011 \cdot 2^2 \approx 1.011$

³Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Beispiele

- Seien die exakten Resultate der Berechnungen
 1. $1.01011 \cdot 2^2$
 2. $1.01010 \cdot 2^2$
 3. $1.01110 \cdot 2^2$
 4. $1.01100 \cdot 2^2$
- ...aber das sind mehr als $p = 4$ Ziffern!³ Wir runden – aber wie?
 1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)

³Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Beispiele

- Seien die exakten Resultate der Berechnungen
 1. $1.01011 \cdot 2^2$
 2. $1.01010 \cdot 2^2$
 3. $1.01110 \cdot 2^2$
 4. $1.01100 \cdot 2^2$
- ...aber das sind mehr als $p = 4$ Ziffern!³ Wir runden – aber wie?
 1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)
 2. $1.01010 \cdot 2^2$

³Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Beispiele

- Seien die exakten Resultate der Berechnungen
 1. $1.01011 \cdot 2^2$
 2. $1.01010 \cdot 2^2$
 3. $1.01110 \cdot 2^2$
 4. $1.01100 \cdot 2^2$
- ...aber das sind mehr als $p = 4$ Ziffern!³ Wir runden – aber wie?
 1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)
 2. $1.01010 \cdot 2^2 \approx 1.010$

³Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Beispiele

- Seien die exakten Resultate der Berechnungen
 1. $1.01011 \cdot 2^2$
 2. $1.01010 \cdot 2^2$
 3. $1.01110 \cdot 2^2$
 4. $1.01100 \cdot 2^2$
- ...aber das sind mehr als $p = 4$ Ziffern!³ Wir runden – aber wie?
 1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)
 2. $1.01010 \cdot 2^2 \approx 1.010$ per (3)

³Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Beispiele

- Seien die exakten Resultate der Berechnungen

1. $1.01011 \cdot 2^2$

2. $1.01010 \cdot 2^2$

3. $1.01110 \cdot 2^2$

4. $1.01100 \cdot 2^2$

- ...aber das sind mehr als $p = 4$ Ziffern!³ Wir runden – aber wie?

1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)

2. $1.01010 \cdot 2^2 \approx 1.010$ per (3)

3. $1.01110 \cdot 2^2$

³Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Beispiele

- Seien die exakten Resultate der Berechnungen

1. $1.01011 \cdot 2^2$

2. $1.01010 \cdot 2^2$

3. $1.01110 \cdot 2^2$

4. $1.01100 \cdot 2^2$

- ...aber das sind mehr als $p = 4$ Ziffern!³ Wir runden – aber wie?

1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)

2. $1.01010 \cdot 2^2 \approx 1.010$ per (3)

3. $1.01110 \cdot 2^2 \approx 1.100$

³Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Beispiele

- Seien die exakten Resultate der Berechnungen
 1. $1.01011 \cdot 2^2$
 2. $1.01010 \cdot 2^2$
 3. $1.01110 \cdot 2^2$
 4. $1.01100 \cdot 2^2$
- ...aber das sind mehr als $p = 4$ Ziffern!³ Wir runden – aber wie?
 1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)
 2. $1.01010 \cdot 2^2 \approx 1.010$ per (3)
 3. $1.01110 \cdot 2^2 \approx 1.100$ per (3)

³Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Beispiele

- Seien die exakten Resultate der Berechnungen

1. $1.01011 \cdot 2^2$

2. $1.01010 \cdot 2^2$

3. $1.01110 \cdot 2^2$

4. $1.01100 \cdot 2^2$

- ...aber das sind mehr als $p = 4$ Ziffern!³ Wir runden – aber wie?

1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)

2. $1.01010 \cdot 2^2 \approx 1.010$ per (3)

3. $1.01110 \cdot 2^2 \approx 1.100$ per (3)

4. $1.01100 \cdot 2^2$

³Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Beispiele

- Seien die exakten Resultate der Berechnungen

1. $1.01011 \cdot 2^2$

2. $1.01010 \cdot 2^2$

3. $1.01110 \cdot 2^2$

4. $1.01100 \cdot 2^2$

- ...aber das sind mehr als $p = 4$ Ziffern!³ Wir runden – aber wie?

1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)

2. $1.01010 \cdot 2^2 \approx 1.010$ per (3)

3. $1.01110 \cdot 2^2 \approx 1.100$ per (3)

4. $1.01100 \cdot 2^2 \approx 1.011$

³Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Beispiele

- Seien die exakten Resultate der Berechnungen

1. $1.01011 \cdot 2^2$

2. $1.01010 \cdot 2^2$

3. $1.01110 \cdot 2^2$

4. $1.01100 \cdot 2^2$

- ...aber das sind mehr als $p = 4$ Ziffern!³ Wir runden – aber wie?

1. $1.01011 \cdot 2^2 \approx 1.011$ per (2)

2. $1.01010 \cdot 2^2 \approx 1.010$ per (3)

3. $1.01110 \cdot 2^2 \approx 1.100$ per (3)

4. $1.01100 \cdot 2^2 \approx 1.011$ per (1)

³Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Fragen/Unklarheiten?

2. Feedback zu **code** expert

Allgemeines bezüglich Feedback/Fragen

Allgemeines bezüglich Feedback/Fragen

- Bitte schickt mir *immer* eine "Follow-up/Reminder"-Mail wenn ihr mir eine Frage im Unterricht gestellt habt und noch eine Antwort erwartet (insb. in Bezug auf die **code expert** -Aufgaben)

Allgemeines bezüglich **code expert**

Allgemeines bezüglich **code expert**

- Bitte behaltet den Code und die Antworten innerhalb der grauen Linie auf **code expert**

Allgemeines bezüglich **code expert**

- Bitte behaltet den Code und die Antworten innerhalb der grauen Linie auf **code expert**
- "There's almost always a better approach", seid nicht traurig, falls euer Ansatz nicht der effizienteste war

Allgemeines bezüglich **code expert**

- Bitte behaltet den Code und die Antworten innerhalb der grauen Linie auf **code expert**
- "There's almost always a better approach", seid nicht traurig, falls euer Ansatz nicht der effizienteste war
- Fast alle Submissions waren viel wortreicher als nötig; fasst euch kürzer

Allgemeines bezüglich **code expert**

- Bitte behaltet den Code und die Antworten innerhalb der grauen Linie auf **code expert**
- "There's almost always a better approach", seid nicht traurig, falls euer Ansatz nicht der effizienteste war
- Fast alle Submissions waren viel wortreicher als nötig; fasst euch kürzer
- Wenn es mehrere ähnliche Aufgaben gab, wurde nur zu einer Aufgabe ein ausführliches Feedback gegeben (und von der zweiten darauf verwiesen)

Allgemeines bezüglich **code expert**

- Bitte behaltet den Code und die Antworten innerhalb der grauen Linie auf **code expert**
- "There's almost always a better approach", seid nicht traurig, falls euer Ansatz nicht der effizienteste war
- Fast alle Submissions waren viel wortreicher als nötig; fasst euch kürzer
- Wenn es mehrere ähnliche Aufgaben gab, wurde nur zu einer Aufgabe ein ausführliches Feedback gegeben (und von der zweiten darauf verwiesen)
- Kein Feedback \implies Gut (genug) gemacht!

Allgemeines bezüglich **code expert**

- Bitte behaltet den Code und die Antworten innerhalb der grauen Linie auf **code expert**
- "There's almost always a better approach", seid nicht traurig, falls euer Ansatz nicht der effizienteste war
- Fast alle Submissions waren viel wortreicher als nötig; fasst euch kürzer
- Wenn es mehrere ähnliche Aufgaben gab, wurde nur zu einer Aufgabe ein ausführliches Feedback gegeben (und von der zweiten darauf verwiesen)
- Kein Feedback \implies Gut (genug) gemacht!
- Ich werde von nun an öfter direkt in eurer Submission Änderungen machen; falls ich auf eine solche Änderung im Code verweise, aber ihr sie nicht sieht, schreibt mir sofort eine Mail

Fragen bezüglich **code expert** eurerseits?

3. Lernziele

Ziele für Heute

Lernziele

Ziele für Heute

Lernziele

- Programme, die Referenzen enthalten, tracen können
- Programme, die `std::vector` verwenden, tracen und schreiben können
- Programme, die mehrdimensionale `std::vector` verwenden, tracen und schreiben können

4. Zusammenfassung

5. Referenzen

Beispiel zu Program Tracing I

```
int a = 3;  
int& b = a;  
  
b = 7;  
  
std::cout << a;
```

Output:

Beispiel zu Program Tracing I

```
int a = 3;  
int& b = a;  
  
b = 7;  
  
std::cout << a;
```

Output: 7

Beispiel zu Program Tracing II

```
void foo(int i){
    i = 5;
}

int main(){
    int i = 4;
    foo(i);
    std::cout << i << std::endl;
}
```

Output:

Beispiel zu Program Tracing II

```
void foo(int i) {  
    i = 5;  
}
```

```
int main() {  
    int i = 4;  
    foo(i);  
    std::cout << i << std::endl;  
}
```

Output: 4 ...aber wieso?

Beispiel zu Program Tracing II

```
void foo(int i){
    i = 5;
}

int main(){
    int i = 4;
    foo(i);
    std::cout << i << std::endl;
}
```

Output: 4 ...aber wieso?

- Referenzen (`type&`) werden als Typ von Funktionsparametern (Inputs) oder Rückgabetypen (Returns) verwendet
- Wenn die Parameter **nicht** *referenced* sind, so sagt man *passed to the function by value* (So haben wir das bei allen bisherigen Funktionen gemacht); dabei wird immer eine Kopie des Inputs für die Funktion angefertigt

Beispiel zu Program Tracing III

```
void foo(int& i){
    i = 5;
}

int main(){
    int i = 4;
    foo(i);
    std::cout << i << std::endl;
}
```

Output:

Beispiel zu Program Tracing III

```
void foo(int& i){  
    i = 5;  
}  
  
int main(){  
    int i = 4;  
    foo(i);  
    std::cout << i << std::endl;  
}
```

Output: 5

Beispiel zu Program Tracing III

```
void foo(int& i){
    i = 5;
}

int main(){
    int i = 4;
    foo(i);
    std::cout << i << std::endl;
}
```

Output: 5

- Wenn ein Funktionsparameter ein Referenztyp (`type&`) ist, sagt man *“passed (the argument) by reference”*

Wieso das Ganze?

Wieso das Ganze?

- da man so mehrere Resultate/Variablen beeinflussen kann und nicht nur auf das `return` angewiesen ist

Wieso das Ganze?

- da man so mehrere Resultate/Variablen beeinflussen kann und nicht nur auf das **return** angewiesen ist
- da man sich so das (teils teure) Kopieren der Parameter spart und somit die Performance des Programms verbessern kann

Wieso das Ganze?

- da man so mehrere Resultate/Variablen beeinflussen kann und nicht nur auf das `return` angewiesen ist
- da man sich so das (teils teure) Kopieren der Parameter spart und somit die Performance des Programms verbessern kann
- da es manchmal einfach nicht anders geht (`std::cout` zum Beispiel werden wir uns in paar Wochen anschauen)

Referenzen als Return Types

Wir haben jetzt Funktionsparameter gesehen, die einen Referenztypen haben, aber auch für return types lassen sich Referenzen verwenden

Referenzen als Return Types

Wir haben jetzt Funktionsparameter gesehen, die einen Referenztypen haben, aber auch für return types lassen sich Referenzen verwenden

```
int& increment(int& m) {  
    return ++m;  
}  
  
int main() {  
    int n = 3;  
  
    increment(increment(n));  
  
    std::cout << n << std::endl;  
}
```

Output:

Referenzen als Return Types

Wir haben jetzt Funktionsparameter gesehen, die einen Referenztypen haben, aber auch für return types lassen sich Referenzen verwenden

```
int& increment(int& m){
    return ++m;
}

int main(){
    int n = 3;

    increment(increment(n));

    std::cout << n << std::endl;
}
```

Output: 5, aber wieso?

Referenzen als Return Types

Wir haben jetzt Funktionsparameter gesehen, die einen Referenztypen haben, aber auch für return types lassen sich Referenzen verwenden

```
int& increment(int& m){
    return ++m;
}

int main(){
    int n = 3;

    increment(increment(n));

    std::cout << n << std::endl;
}
```

Output: 5, aber wieso? Wegen der Referenz im return type!

Fragen/Unklarheiten?

Referenz oder Kopie? I

```
int foo(int& a, int b){
    a += b;
    return a;
}

int main(){
    int a = 0;
    int b = 1;
    for(int i = 0; i < 5; ++i){
        b = foo(a, b);
        std::cout << b << " ";
    }
    return 0;
}
```

a	0	1	2	...
b	1	1	2	...
i				...

Output: 1

Referenz oder Kopie? I

```
int foo(int& a, int b){
    a += b;
    return a;
}

int main(){
    int a = 0;
    int b = 1;
    for(int i = 0; i < 5; ++i){
        b = foo(a, b);
        std::cout << b << " ";
    }
    return 0;
}
```

Output: 1 2 4 8 16

Referenz oder Kopie? II

```
int foo(int a, int b){
    a += b;
    return a;
}

int main(){
    int a = 0;
    int b = 1;
    for(int i = 0; i < 5; ++i){
        b = foo(a, b);
        std::cout << b << " ";
    }
    return 0;
}
```

Output:

Referenz oder Kopie? II

```
int foo(int a, int b){
    a += b;
    return a;
}

int main(){
    int a = 0;
    int b = 1;
    for(int i = 0; i < 5; ++i){
        b = foo(a, b);
        std::cout << b << " ";
    }
    return 0;
}
```

Output: 1 1 1 1 1

Referenz oder Kopie? III

```
int foo(int a, int& b){
    a += b;
    return a;
}

int main(){
    int a = 0;
    int b = 1;
    for(int i = 0; i < 5; ++i){
        b = foo(a, b);
        std::cout << b << " ";
    }
    return 0;
}
```

Output:

Referenz oder Kopie? III

```
int foo(int a, int& b){
    a += b;
    return a;
}

int main(){
    int a = 0;
    int b = 1;
    for(int i = 0; i < 5; ++i){
        b = foo(a, b);
        std::cout << b << " ";
    }
    return 0;
}
```

Output: 1 1 1 1 1

cf. script p. 16

Fragen/Unklarheiten?

6. `std::vector<T>`

std::vector, aber wie?

- `#include <vector>`

std::vector, aber wie?

- #include <vector>

- Vektoren kann man sich als eine Reihe von Kästchen vorstellen, die je einen Wert des gegebenen Typs speichern

`std::vector<T>` → `int, double,`
`std::vector<T>`



std::vector, aber wie?

- `#include <vector>`
- Vektoren kann man sich als eine Reihe von Kästchen vorstellen, die je einen Wert des gegebenen Typs speichern
- Man kann Vektoren etwa so behandeln wie einen neuen Typen

std::vector, aber wie?

- `#include <vector>`
- Vektoren kann man sich als eine Reihe von Kästchen vorstellen, die je einen Wert des gegebenen Typs speichern
- Man kann Vektoren etwa so behandeln wie einen neuen Typen
- Es gibt viele Möglichkeiten, einen Vector zu initialisieren/definieren. Schaut im Vorlesungsmaterial nach oder sucht online

std::vector, aber wie?

- `#include <vector>`
- Vektoren kann man sich als eine Reihe von Kästchen vorstellen, die je einen Wert des gegebenen Typs speichern
- Man kann Vektoren etwa so behandeln wie einen neuen Typen
- Es gibt viele Möglichkeiten, einen Vector zu initialisieren/definieren. Schaut im Vorlesungsmaterial nach oder sucht online
- `myvector[n-1]`
um den n -ten Wert im Vektor `myvector` zu benutzen

std::vector, aber wie?

- `#include <vector>`
- Vektoren kann man sich als eine Reihe von Kästchen vorstellen, die je einen Wert des gegebenen Typs speichern
- Man kann Vektoren etwa so behandeln wie einen neuen Typen
- Es gibt viele Möglichkeiten, einen Vector zu initialisieren/definieren. Schaut im Vorlesungsmaterial nach oder sucht online
- `myvector[n-1]`
um den n -ten Wert im Vektor `myvector` zu benutzen
- `myvector.push_back(x)`
um den Wert `x` anzuhängen



Fragen/Unklarheiten?

Übung: "Reversing Vectors"

Let's code together!

Code Example "Reversing Vectors" auf [code expert](#)

Übung: "Reversing Vectors" – Beispiellösung

```
// POST: Prints a vector in reverse without side-effects
void efficient_reverse_print(std::vector<int>& sequence) {

    for (int i = sequence.size() - 1; i >= 0; i--) {
        std::cout << sequence[i] << " ";
    }

    std::cout << std::endl;

}
```

7. Multidimensionale Vektoren

Was sind Multidimensionale Vektoren?

Multidimensionale Vektoren sind Matrizen⁴

```
matrix.at(row_index)           // Accessing vector<T> (entire row)
matrix.at(row_index).at(col_index) // Accessing T (single element)
```

⁴eigentlich sind sie Vektoren von Vektoren!

Aufgabe "Matrix Transpose"

- Öffnet "Matrix Transpose" auf **code expert**

Aufgabe "Matrix Transpose"

- Öffnet "Matrix Transpose" auf **code expert**

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet

Aufgabe "Matrix Transpose"

- Öffnet "Matrix Transpose" auf **code expert**

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet
- Vereinfachung der Syntax:

```
using irow = std::vector<int>;  
using imatrix = std::vector<irow>;
```

Aufgabe "Matrix Transpose"

- Öffnet "Matrix Transpose" auf **code expert**

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet
- Vereinfachung der Syntax:

```
using irow = std::vector<int>;  
using imatrix = std::vector<irow>;
```
- Programmiert eine Lösung (optional in Gruppen)

Lösung zu "Matrix Transpose"

Lösung zu "Matrix Transpose"

```
imatrix transpose_matrix(const imatrix& matrix) {
    int rows = get_rows(matrix);
    int cols = get_cols(matrix);

    // construct a matrix with zero rows
    imatrix transposed_matrix = imatrix(0);
    for (int col_index = 0; col_index < cols; col_index++) {
        // construct a row with zero entries
        irow row = irow(0);
        for (int row_index = 0; row_index < rows; row_index++) {
            row.push_back(matrix[row_index][col_index]);
        }
        transposed_matrix.push_back(row);
    }
    return transposed_matrix;
}
```

Fragen/Unklarheiten?

8. Repetition: Fließkommazahlen

Aufgabe

- Versucht, folgende Aufgaben zu lösen
- Fragt, wenn etwas unklar ist

Informatik
Exercise Session

Consider the normalized floating point number system $F^*(\beta, p, e_{\min}, e_{\max})$ with $\beta = 2$, $p = 3$, $e_{\min} = -4$, $e_{\max} = 4$.

Compute the following expressions as the parentheses suggest, representing each intermediate result (and the final result) in the normalized floating point system according to the rules of computing with floating point numbers.

(10 + 0.5) + 0.5			(0.5 + 0.5) + 10		
	decimal	binary		decimal	binary
	10	?????		0.5	?????
+	0.5	?????	+	0.5	?????
=		?????	=		?????
+	0.5	?????	+	10	?????
=	??	← ?????	=	??	← ?????

$(10 + 0.5) + 0.5$			$(0.5 + 0.5) + 10$		
decimal		binary	decimal		binary
10		$1.01 \cdot 2^3$	0.5		?????
+ 0.5		$0.0001 \cdot 2^3$	+ 0.5		?????
=		?????	=		?????
+ 0.5		?????	+ 10		?????
= ??	←	?????	= ??	←	?????

$(10 + 0.5) + 0.5$			$(0.5 + 0.5) + 10$		
decimal		binary	decimal		binary
10		$1.01 \cdot 2^3$	0.5		?????
+ 0.5		$0.0001 \cdot 2^3$	+ 0.5		?????
=		$1.0101 \cdot 2^3$	=		?????
+ 0.5		?????	+ 10		?????
= ??	←	?????	= ??	←	?????

$(10 + 0.5) + 0.5$			$(0.5 + 0.5) + 10$		
decimal		binary	decimal		binary
10		$1.01 \cdot 2^3$	0.5		?????
+ 0.5		$0.0001 \cdot 2^3$	+ 0.5		?????
=		$1.01 \cdot 2^3$	=		?????
+ 0.5		$0.0001 \cdot 2^3$	+ 10		?????
= ??	←	?????	= ??	←	?????

$(10 + 0.5) + 0.5$			$(0.5 + 0.5) + 10$		
decimal		binary	decimal		binary
10		$1.01 \cdot 2^3$	0.5		?????
+ 0.5		$0.0001 \cdot 2^3$	+ 0.5		?????
=		$1.01 \cdot 2^3$	=		?????
+ 0.5		$0.0001 \cdot 2^3$	+ 10		?????
= 10	←	$1.01 \cdot 2^3$	= ??	←	?????

$(10 + 0.5) + 0.5$			$(0.5 + 0.5) + 10$		
decimal		binary	decimal		binary
10		$1.01 \cdot 2^3$	0.5		$1.00 \cdot 2^{-1}$
+ 0.5		$0.0001 \cdot 2^3$	+ 0.5		$1.00 \cdot 2^{-1}$
=		$1.01 \cdot 2^3$	=		?????
+ 0.5		$0.0001 \cdot 2^3$	+ 10		?????
= 10	←	$1.01 \cdot 2^3$	= ??	←	?????

$(10 + 0.5) + 0.5$		$(0.5 + 0.5) + 10$	
decimal	binary	decimal	binary
10	$1.01 \cdot 2^3$	0.5	$1.00 \cdot 2^{-1}$
+ 0.5	$0.0001 \cdot 2^3$	+ 0.5	$1.00 \cdot 2^{-1}$
=	$1.01 \cdot 2^3$	=	$1.00 \cdot 2^0$
+ 0.5	$0.0001 \cdot 2^3$	+ 10	$1010.00 \cdot 2^0$
= 10	← $1.01 \cdot 2^3$	= ??	← ?????

$(10 + 0.5) + 0.5$		$(0.5 + 0.5) + 10$	
decimal	binary	decimal	binary
10	$1.01 \cdot 2^3$	0.5	$1.00 \cdot 2^{-1}$
+ 0.5	$0.0001 \cdot 2^3$	+ 0.5	$1.00 \cdot 2^{-1}$
=	$1.01 \cdot 2^3$	=	$1.00 \cdot 2^0$
+ 0.5	$0.0001 \cdot 2^3$	+ 10	$1010.00 \cdot 2^0$
= 10	$\leftarrow 1.01 \cdot 2^3$	= ??	$\leftarrow 1011.00 \cdot 2^0$

$(10 + 0.5) + 0.5$		$(0.5 + 0.5) + 10$	
decimal	binary	decimal	binary
10	$1.01 \cdot 2^3$	0.5	$1.00 \cdot 2^{-1}$
+ 0.5	$0.0001 \cdot 2^3$	+ 0.5	$1.00 \cdot 2^{-1}$
=	$1.01 \cdot 2^3$	=	$1.00 \cdot 2^0$
+ 0.5	$0.0001 \cdot 2^3$	+ 10	$1010.00 \cdot 2^0$
= 10	← $1.01 \cdot 2^3$	= ??	← $1.011 \cdot 2^3$

$(10 + 0.5) + 0.5$		$(0.5 + 0.5) + 10$	
decimal	binary	decimal	binary
10	$1.01 \cdot 2^3$	0.5	$1.00 \cdot 2^{-1}$
+ 0.5	$0.0001 \cdot 2^3$	+ 0.5	$1.00 \cdot 2^{-1}$
=	$1.01 \cdot 2^3$	=	$1.00 \cdot 2^0$
+ 0.5	$0.0001 \cdot 2^3$	+ 10	$1010.00 \cdot 2^0$
= 10	$\leftarrow 1.01 \cdot 2^3$	= 12	$\leftarrow 1.10 \cdot 2^3$

Fragen/Unklarheiten?

9. Alte Prüfungsfragen

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem⁵

$$F^*(\beta = 2, p = 3, e_{\min} = -1, e_{\max} = 4)$$

Richtig oder Falsch?

⁵Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem⁵

$$F^*(\beta = 2, p = 3, e_{\min} = -1, e_{\max} = 4)$$

Richtig oder Falsch?

1. "1.25 kann im Fließkommazahlensystem exakt dargestellt werden"

⁵Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem⁵

$$F^*(\beta = 2, p = 3, e_{\min} = -1, e_{\max} = 4)$$

Richtig oder Falsch?

1. "1.25 kann im Fließkommazahlensystem exakt dargestellt werden"
Richtig, nämlich $1.01 \cdot 2^0$

⁵Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem⁵

$$F^*(\beta = 2, p = 3, e_{\min} = -1, e_{\max} = 4)$$

Richtig oder Falsch?

1. "1.25 kann im Fließkommazahlensystem exakt dargestellt werden"
Richtig, nämlich $1.01 \cdot 2^0$
2. "Es existiert keine Zahl $Z \in F^*$, für die gilt: $0.0625 < Z < 0.25$ "

⁵Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem⁵

$$F^*(\beta = 2, p = 3, e_{\min} = -1, e_{\max} = 4)$$

Richtig oder Falsch?

1. "1.25 kann im Fließkommazahlensystem exakt dargestellt werden"
Richtig, nämlich $1.01 \cdot 2^0$
2. "Es existiert keine Zahl $Z \in F^*$, für die gilt: $0.0625 < Z < 0.25$ "
Richtig, die kleinste darstellbare Zahl ist 0.5 (i.e., $1.0 * 2^{-1}$)

⁵Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem⁵

$$F^*(\beta = 2, p = 3, e_{\min} = -1, e_{\max} = 4)$$

Richtig oder Falsch?

1. "1.25 kann im Fließkommazahlensystem exakt dargestellt werden"
Richtig, nämlich $1.01 \cdot 2^0$
2. "Es existiert keine Zahl $Z \in F^*$, für die gilt: $0.0625 < Z < 0.25$ "
Richtig, die kleinste darstellbare Zahl ist 0.5 (i.e., $1.0 * 2^{-1}$)
3. "3.25 kann genau in F^* dargestellt werden"

⁵Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem⁵

$$F^*(\beta = 2, p = 3, e_{\min} = -1, e_{\max} = 4)$$

Richtig oder Falsch?

1. "1.25 kann im Fließkommazahlensystem exakt dargestellt werden"
Richtig, nämlich $1.01 \cdot 2^0$
2. "Es existiert keine Zahl $Z \in F^*$, für die gilt: $0.0625 < Z < 0.25$ "
Richtig, die kleinste darstellbare Zahl ist 0.5 (i.e., $1.0 * 2^{-1}$)
3. "3.25 kann genau in F^* dargestellt werden"
Falsch, $3.25 = 1,101 * 2^1$ würde die Genauigkeit $p \geq 4$ erfordern

⁵Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage "Schleife"

```
int sum = 17;
int i = 1;

do {
    i += sum;
    sum = sum / 2;
} while (i > sum && sum >= 0);

std::cout << sum;
```

Welche Aussage beschreibt den Output am besten?

Prüfungsfrage "Schleife"

```
int sum = 17;
int i = 1;

do {
    i += sum;
    sum = sum / 2;
} while (i > sum && sum >= 0);

std::cout << sum;
```

Welche Aussage beschreibt den Output am besten?

- 17
- 8
- Terminiert nie
- 18

Prüfungsfrage "Loop Termination"

```
int sum = 17;
int i = 1;

do {
    i += sum;
    sum = sum / 2;
} while (i > sum && sum >= 0);

std::cout << sum;
```

Antwort:

Prüfungsfrage "Loop Termination"

```
int sum = 17;
int i = 1;

do {
    i += sum;
    sum = sum / 2;
} while (i > sum && sum >= 0);

std::cout << sum;
```

Antwort: Es terminiert nie!

Prüfungsfrage "Loop Termination"

```
int sum = 17;
int i = 1;

do {
    i += sum;
    sum = sum / 2;
} while (i > sum && sum >= 0);

std::cout << sum;
```

Antwort: Es terminiert nie!

- Division von zwei positiven `int` kann nicht negativ sein
⇒ `sum >= 0` ist immer wahr!
- Nach der ersten Ausführung des do-Blocks: `i > sum`.
`sum` ist monoton fallend, `i` ist monoton steigend
⇒ `i > sum` ist immer wahr.

10. Outro

Allgemeine Fragen?

Bis zum nächsten Mal

Schöne Woche noch!