

Übungsstunde — Informatik — ES01

Adel Gavranović

Organisatorisches, Integer Division & Modulo, Binärrepräsentation,
Expressions und Evaluationen

Übersicht

Kennenlernen
Organisatorisches
Zusammenfassung
Integer Division & Modulo
Binärrepräsentation
Expressions und Evaluationen
Binärrepräsentation von neg. Integern
Tipps zu **code expert**
Outro



`n.ethz.ch/~agavranovic`

[Exercise Session Material](#)

[Adel's Webpage](#)

[Mail to Adel](#)

Herzlich Willkommen!

Ziele für heute

- Uns kennenlernen
- Organisatorisches besprechen
- code expert** einrichten
- Int-Division, Operatoren und Modulo verstehen
- Zahlen in andere Basen umrechnen können
- Valide C++-Expressions erkennen
- C++-Expressions evaluieren können

1. Kennenlernen

Adel

- Hatte vor der ETH praktisch keine Erfahrungen im Programmieren

Adel

- Hatte vor der ETH praktisch keine Erfahrungen im Programmieren
- Hatte gute Note in Informatik am BPB-I

Adel

- Hatte vor der ETH praktisch keine Erfahrungen im Programmieren
- Hatte gute Note in Informatik am BPB-I
- Wird dafür sorgen, dass Ihr den BPB-I nicht wegen Informatik verkackt

Adel

- Hatte vor der ETH praktisch keine Erfahrungen im Programmieren
- Hatte gute Note in Informatik am BPB-I
- Wird dafür sorgen, dass Ihr den BPB-I nicht wegen Informatik verkackt
- Hier, um euch zu helfen

Euer Assistent

Adel

- Hatte vor der ETH praktisch keine Erfahrungen im Programmieren
- Hatte gute Note in Informatik am BPB-I
- Wird dafür sorgen, dass Ihr den BPB-I nicht wegen Informatik verkackt
- Hier, um euch zu helfen
- Trinkt gern guten Kaffee
- Fährt gern Fahrrad
- Aus Züri



Eure Kommilitoninnen und Kommilitonen

Stell dich vor!

Eure Kommilitoninnen und Kommilitonen

Stell dich vor!

- Wie heisst du?
- Woher kommst du?
- Wieso RW an der ETH?
- Hast du Erfahrung im Programmieren (insb. C++)?
- Hast du coole Hobbies oder aussergewöhnliche Interessen?

Fragen/Unklarheiten?

2. Organisatorisches



Informatik

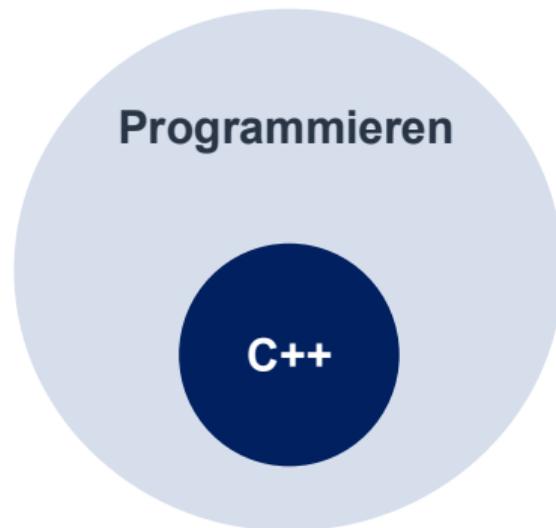
Informationen zur Organisation

Kennenlernen

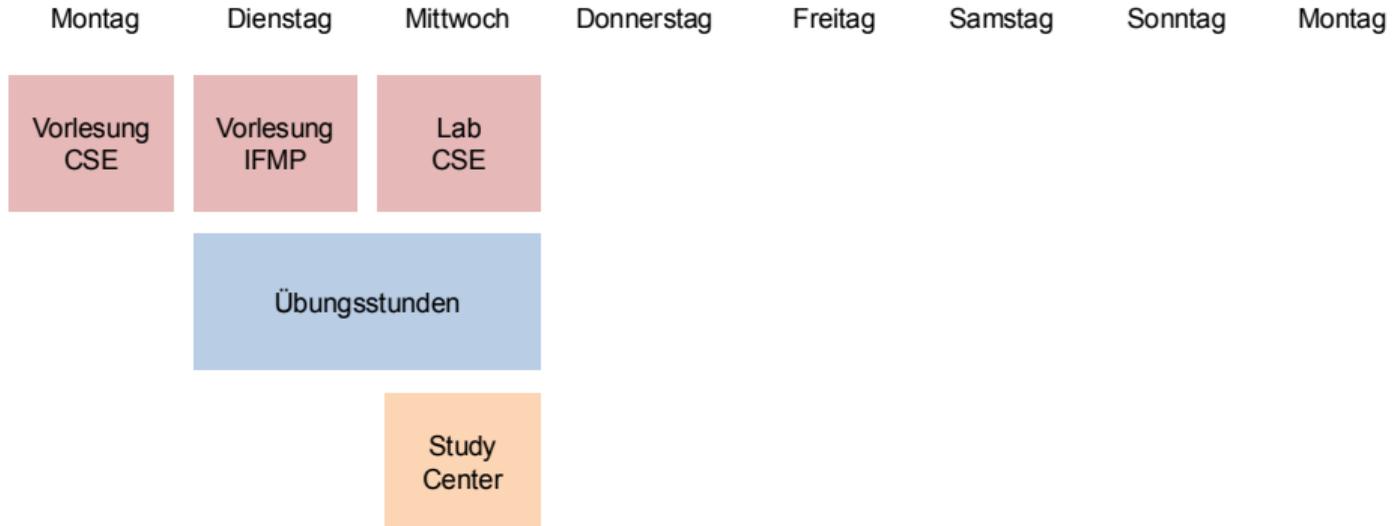
- Name?
- Welcher Teilbereich deines Studiums interessiert dich am meisten?

Ziel des Kurses

- Vorlesung
- Übungsstunde
- Wöchentliche Übungen
- Bonusaufgaben
- Study Center



Wöchentlicher Plan



Hausaufgaben auf CodeExpert lösen
(Ausgabe am Montag 6:00, Abgabe am folgenden Montag 18:00)

Wöchentliche Übungen und Bonusaufgaben

- Alle Übungen findet ihr auf [code]expert (<https://expert.ethz.ch>).
- Ihr müsst euch zuerst mithilfe dem euch gesandten Link in die Übungsgruppe einschreiben.

- **Wöchentliche Übungen:**
 - Zweck: Neuen Stoff anwenden.
 - Veröffentlichung: Montag um 6:00.
 - Deadline: Eine Woche später (Montag 18:00).
 - Erlaubt euch Erfahrungspunkte (XP) zu sammeln.

- **Bonusaufgaben** (benötigt ca. 2/3 der Erfahrungspunkte zur Freischaltung):
 - Zweck: Wissen zu verschiedenen Themen kombinieren.
 - Erlaubt euch max. +0.25 Bonus für die Endnote zu erreichen (mit 2/3 der Bonuspunkte).

Wöchentliche Übungen und Bonusaufgaben

- Alle Übungen findet ihr auf [code]expert (<https://expert.ethz.ch>).
- Ihr müsst euch für eine Übungsaufgabe in einer Übungsaufgaben-Gruppe einschreiben.
- **Wichtig!** Benutzt zum Lösen der Übungsaufgaben **nur** das, was bereits in der Vorlesung vorgestellt wurde und was nicht von der Aufgabenstellung verboten wird. Beachtet, dass der Autograder weitere Einschränkungen (wie die Vermeidung von globalen Variablen) auferlegen kann, die nicht ausdrücklich in der Aufgabenstellung angegeben sind. Beachtet außerdem, dass Warnungen als Fehler behandelt werden. Überprüft also die Ausgabe des Autograders sorgfältig, um zu vermeiden, dass ihr 0 Punkte bekommt. Jede Woche wird eine Zusammenfassung der in dieser Woche vorgestellten Konzepte herausgegeben.
- **Wöchentliche Bonusaufgaben** (Zusammenfassung der Konzepte):
 - Zweck: Neben den wöchentlichen Übungen soll das Wissen zu verschiedenen Themen kombiniert werden.
 - Veröffentlicht: Die Aufgabenstellungen sind jeweils am Montag vor der Vorlesung zu sehen.
 - Deadline: Die Aufgabenstellungen sind jeweils am Montag vor der Vorlesung zu sehen.
 - Erlaubt euch max. +0.25 Bonus für die Endnote zu erreichen (mit 2/3 der Bonuspunkte).

Übungsstunden

- Zweck: Vorbereitung für das Lösen von zukünftigen und vergangenen Übungen.
- Ansatz: Vor allem interaktiver Unterricht und konstruktive Diskussionen.
- Wir erwarten, dass ihr:
 - Aktiv am Unterricht teilnehmt
 - Fragen stellt, wenn ihr den Inhalt nicht versteht, oder warum wir ein bestimmtes Thema behandeln
- Es ist völlig normal, wenn ihr Fehler macht, wir sind im Lernprozess. Bitte vermeidet es, Sachen zu tun, welche andere stören könnten. Wenn eine Aufgabe zu einfach ist, helft anderen.

Study Center

- Zweck: Eine Möglichkeit, um nach individueller Hilfe zum Kurs zu fragen

IFMP:

- Zeit: Mittwoch 16:15-18:15, beginnend ab dem 25. September
- Ort: Mensa Polyterrasse
- Link: <https://studycenter.ethz.ch/>

CSE (mit MAVT geteilt):

- Zeit: Mittwoch 18:15-20:00, beginnend ab dem 2. Oktober
- Ort: ETA F 5

Informationen & Kontakt

- Mehr Informationen sind auf dem Informationsblatt zur Organisation zu finden.
- Für Fragen betreffend der *Vorlesungsinhalte* könnt ihr während der Vorlesung fragen.
- Für Fragen betreffend der *Übungen* könnt ihr euren TA fragen.
- Für *administrative* Fragen kontaktiert bitte den Head-TA (siehe Website für die E-Mail-Adresse).

Fragen/Unklarheiten?

Zur Übungsstunde

- Alles wird auf der meiner Website hochgeladen

Zur Übungsstunde

- Alles wird auf der meiner Website hochgeladen
- Wird *nicht* aufgezeichnet

Zur Übungsstunde

- Alles wird auf der meiner Website hochgeladen
- Wird *nicht* aufgezeichnet
- Bitte stellt Fragen bei Unklarheiten
- Bitte beteiligt euch am Unterricht
- Bitte korrigiert mich, sobald ich Fehler mache

Zur Übungsstunde

- Alles wird auf der meiner Website hochgeladen
- Wird *nicht* aufgezeichnet
- Bitte stellt Fragen bei Unklarheiten
- Bitte beteiligt euch am Unterricht
- Bitte korrigiert mich, sobald ich Fehler mache
- Ihr könnt Fragen auch direkt in euren Code bei **code expert** schreiben, aber das sehe ich erst *nach* der Abgabe

```
| int a = 42; // Eure Kommentare und Fragen hierhin
```

- Fragen via Mail sind immer willkommen

Zum Kurs

- Ist nicht der schwierigste Kurs...

Zum Kurs

- Ist nicht der schwierigste Kurs...
- ...aber einer der wichtigsten

Zum Kurs

- Ist nicht der schwierigste Kurs...
- ...aber einer der wichtigsten
- Falls ihr den BPB-I nicht besteht, wird es wahrscheinlich nicht an Informatik liegen

Zum Kurs

- Ist nicht der schwierigste Kurs...
- ...aber einer der wichtigsten
- Falls ihr den BPB-I nicht besteht, wird es wahrscheinlich nicht an Informatik liegen
- Prüfungen werden erst im Januar stattfinden, aber...

Zum Kurs

- Ist nicht der schwierigste Kurs...
- ...aber einer der wichtigsten
- Falls ihr den BPB-I nicht besteht, wird es wahrscheinlich nicht an Informatik liegen
- Prüfungen werden erst im Januar stattfinden, aber...
- **...ÜBT WÄHREND DES SEMESTERS UND BLEIBT DRAN**

Zum Kurs

- Ist nicht der schwierigste Kurs...
- ...aber einer der wichtigsten
- Falls ihr den BPB-I nicht besteht, wird es wahrscheinlich nicht an Informatik liegen
- Prüfungen werden erst im Januar stattfinden, aber...
- **...ÜBT WÄHREND DES SEMESTERS UND BLEIBT DRAN**
- ...und falls ihr doch mal abhängt: besorgt euch Hilfe, z.B. bei
 - mir
 - dem Study Center
 - euren Mitstudierenden
 - der Kanzlei

How to **code** expert

(öffnet `expert.ethz.ch` auf dem Gerät eurer Wahl)

How to **code expert**

(öffnet `expert.ethz.ch` auf dem Gerät eurer Wahl)

- **code expert** kann ein wenig pingelig sein

How to **code expert**

(öffnet `expert.ethz.ch` auf dem Gerät eurer Wahl)

- **code expert** kann ein wenig pingelig sein
- deshalb folgt den Anweisungen sehr genau (vermeidet unnötigen Text)

How to **code expert**

(öffnet `expert.ethz.ch` auf dem Gerät eurer Wahl)

- **code expert** kann ein wenig pingelig sein
- deshalb folgt den Anweisungen sehr genau (vermeidet unnötigen Text)
- der Autograder wird den Grossteil der Korrekturen übernehmen
- von mir bekommt ihr die letzten paar Punkte für Stil, Dokumentation, Herangehensweise, etc.

How to **code expert**

(öffnet `expert.ethz.ch` auf dem Gerät eurer Wahl)

- **code expert** kann ein wenig pingelig sein
- deshalb folgt den Anweisungen sehr genau (vermeidet unnötigen Text)
- der Autograder wird den Grossteil der Korrekturen übernehmen
- von mir bekommt ihr die letzten paar Punkte für Stil, Dokumentation, Herangehensweise, etc.
- Textaufgaben werden vollständig manuell korrigiert

How to **code expert**

(öffnet `expert.ethz.ch` auf dem Gerät eurer Wahl)

- **code expert** kann ein wenig pingelig sein
- deshalb folgt den Anweisungen sehr genau (vermeidet unnötigen Text)
- der Autograder wird den Grossteil der Korrekturen übernehmen
- von mir bekommt ihr die letzten paar Punkte für Stil, Dokumentation, Herangehensweise, etc.
- Textaufgaben werden vollständig manuell korrigiert
 - Löst Textaufgaben bitte mit Markdown+ \LaTeX in **code expert**

How to **code expert**

(öffnet `expert.ethz.ch` auf dem Gerät eurer Wahl)

- **code expert** kann ein wenig pingelig sein
- deshalb folgt den Anweisungen sehr genau (vermeidet unnötigen Text)
- der Autograder wird den Grossteil der Korrekturen übernehmen
- von mir bekommt ihr die letzten paar Punkte für Stil, Dokumentation, Herangehensweise, etc.
- Textaufgaben werden vollständig manuell korrigiert
 - Löst Textaufgaben bitte mit Markdown+ \LaTeX in **code expert**
 - Schlimmstenfalls ladet ein lesbares PDF direkt auf **code expert** hoch

How to **code expert**

(öffnet `expert.ethz.ch` auf dem Gerät eurer Wahl)

- **code expert** kann ein wenig pingelig sein
- deshalb folgt den Anweisungen sehr genau (vermeidet unnötigen Text)
- der Autograder wird den Grossteil der Korrekturen übernehmen
- von mir bekommt ihr die letzten paar Punkte für Stil, Dokumentation, Herangehensweise, etc.
- Textaufgaben werden vollständig manuell korrigiert
 - Löst Textaufgaben bitte mit Markdown+ \LaTeX in **code expert**
 - Schlimmstenfalls ladet ein lesbares PDF direkt auf **code expert** hoch
- Feedback zu den Aufgaben könnt ihr innerhalb 10 Tagen erwarten (falls dringend, einfach eine Mail an mich)

Fragen/Unklarheiten?

3. Zusammenfassung

Zusammenfassung

Zusammenfassung

Jede Woche erscheint auf der Kurswebseite¹ eine Zusammenfassung (“Summary”) des Stoffs der vorherige Woche. Diese könnt ihr beim Lösen der Aufgaben benutzen.

¹  Kurswebseite

Zusammenfassung

Jede Woche erscheint auf der Kurswebseite¹ eine Zusammenfassung (“Summary”) des Stoffs der vorherige Woche. Diese könnt ihr beim Lösen der Aufgaben benutzen.

Frage: Wollen wir diese jede Woche gemeinsam anschauen?

¹  Kurswebseite

4. Integer Division & Modulo

Der Unterschied zwischen = und ==

Der Unterschied zwischen = und ==

Assignment Operator (=)

Der Unterschied zwischen = und ==

Assignment Operator (=)

gebraucht, um Variablen Werte zuzuweisen

Der Unterschied zwischen = und ==

Assignment Operator (=)

gebraucht, um Variablen Werte zuzuweisen

```
int a = 42; // assigns the value 42 to the variable a  
int b = 18; // assigns the value 18 to the variable b
```

Equality Operator (==)

Der Unterschied zwischen = und ==

Assignment Operator (=)

gebraucht, um Variablen Werte zuzuweisen

```
int a = 42; // assigns the value 42 to the variable a  
int b = 18; // assigns the value 18 to the variable b
```

Equality Operator (==)

gebraucht, um Gleichheit zwischen Variablen zu überprüfen

Der Unterschied zwischen = und ==

Assignment Operator (=)

gebraucht, um Variablen Werte zuzuweisen

```
int a = 42; // assigns the value 42 to the variable a  
int b = 18; // assigns the value 18 to the variable b
```

Equality Operator (==)

gebraucht, um Gleichheit zwischen Variablen zu überprüfen

```
(a == b) // this "expression" will equal 1 (true)  
         // or 0 (false) ("boolean")
```

Integer Division & Modulo

Integer Division & Modulo

Integer Division (a/b)

Der Compiler “ignoriert”
Dezimalstellen, wenn er (`unsigned`)
`int` durch (`unsigned`) `int` teilt.

Integer Division & Modulo

Integer Division (a/b)

Der Compiler "ignoriert"
Dezimalstellen, wenn er (`unsigned`)
`int` durch (`unsigned`) `int` teilt.

Modulo (a%b)

Division mit Rest, aber *nur* den Rest.

Integer Division & Modulo

Integer Division (a/b)

Der Compiler “ignoriert”
Dezimalstellen, wenn er (**unsigned**)
int durch (**unsigned**) **int** teilt.

7/3 ==

Modulo (a%b)

Division mit Rest, aber *nur* den Rest.

Integer Division & Modulo

Integer Division (a/b)

Der Compiler “ignoriert”
Dezimalstellen, wenn er (**unsigned**)
int durch (**unsigned**) **int** teilt.

7/3 == 2

15/4 ==

Modulo (a%b)

Division mit Rest, aber *nur* den Rest.

Integer Division & Modulo

Integer Division (a/b)

Der Compiler “ignoriert”
Dezimalstellen, wenn er (**unsigned**)
int durch (**unsigned**) **int** teilt.

7/3 == 2

15/4 == 3

16/4 ==

Modulo (a%b)

Division mit Rest, aber *nur* den Rest.

Integer Division & Modulo

Integer Division (a/b)

Der Compiler "ignoriert" Dezimalstellen, wenn er (**unsigned**) **int** durch (**unsigned**) **int** teilt.

$$7/3 == 2$$

$$15/4 == 3$$

$$16/4 == 4$$

Modulo (a%b)

Division mit Rest, aber *nur* den Rest.

$$7\%3 ==$$

Integer Division & Modulo

Integer Division (a/b)

Der Compiler "ignoriert" Dezimalstellen, wenn er (**unsigned**) **int** durch (**unsigned**) **int** teilt.

$$7/3 == 2$$

$$15/4 == 3$$

$$16/4 == 4$$

Modulo (a%b)

Division mit Rest, aber *nur* den Rest.

$$7\%3 == 1$$

$$15\%4 ==$$

Integer Division & Modulo

Integer Division (a/b)

Der Compiler "ignoriert" Dezimalstellen, wenn er (**unsigned**) **int** durch (**unsigned**) **int** teilt.

$$7/3 == 2$$

$$15/4 == 3$$

$$16/4 == 4$$

Modulo (a%b)

Division mit Rest, aber *nur* den Rest.

$$7\%3 == 1$$

$$15\%4 == 3$$

$$16\%4 ==$$

Integer Division & Modulo

Integer Division (a/b)

Der Compiler "ignoriert" Dezimalstellen, wenn er (**unsigned**) **int** durch (**unsigned**) **int** teilt.

$$7/3 == 2$$

$$15/4 == 3$$

$$16/4 == 4$$

Modulo (a%b)

Division mit Rest, aber *nur* den Rest.

$$7\%3 == 1$$

$$15\%4 == 3$$

$$16\%4 == 0$$

Integer Division & Modulo

Integer Division (a/b)

Der Compiler "ignoriert" Dezimalstellen, wenn er (**unsigned**) **int** durch (**unsigned**) **int** teilt.

$$7/3 == 2$$

$$15/4 == 3$$

$$16/4 == 4$$

Modulo (a%b)

Division mit Rest, aber *nur* den Rest.

$$7\%3 == 1$$

$$15\%4 == 3$$

$$16\%4 == 0$$

Wichtige Identität

$$(a / b) * b + a \% b == a$$

Fragen/Unklarheiten?

Code Snippet Quiz

Frage:

Code Snippet Quiz

Frage: Welche Frage beantwortet dieses Code Snippet?

```
int a;
std::cin >> a;
if (a % 2 == 0) {
    std::cout << "Yes" << std::endl;
} else {
    std::cout << "No" << std::endl;
}
```

Antwort:

Code Snippet Quiz

Frage: Welche Frage beantwortet dieses Code Snippet?

```
int a;
std::cin >> a;
if (a % 2 == 0) {
    std::cout << "Yes" << std::endl;
} else {
    std::cout << "No" << std::endl;
}
```

Antwort: Die Ausgabe beantwortet, ob die eingegebene Zahl (die in a gespeichert wird) eine gerade Zahl ist.

Mal schauen, was ihr gelernt habt

- Geht auf `expert.ethz.ch`
- Logt euch ein
- Geht zu “Code Examples”
- Unter “Lecture 2: Exercise Session”, öffnet “Last Three Digits”

Mal schauen, was ihr gelernt habt

- Geht auf `expert.ethz.ch`
- Logt euch ein
- Geht zu “Code Examples”
- Unter “Lecture 2: Exercise Session”, öffnet “Last Three Digits”
- Versucht die Aufgabe zu lösen (10 Minuten)
- Wir schauen uns eure Ansätze später an

Aufgabe

Task “Last Three Digits”

Write a program which reads in an integer a larger than 1000 and outputs its last three digits with a space between them.

For example, if $a = 14325$, the output should be 3 2 5.

Lösung

```
#include <iostream>
int main() {
    int a;
    // input
    std::cin >> a;
    // computation
    int digit0 = a % 10;
    int remainder0 = a / 10;
    int digit1 = remainder0 % 10;
    int remainder1 = remainder0 / 10;
    int digit2 = remainder1 % 10;
    // output
    std::cout << digit2 << " " << digit1 << " " << digit0 << std::endl;
    return 0;
}
```

5. Binärrepräsentation

Binärrepräsentation

Binärrepräsentation

0	1	0	0	0	1	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
x128	x64	x32	x16	x8	x4	x2	x1

$$+ 2^6 + \dots + 4 + 2 + 0$$

Binärrepräsentation

0	1	0	0	0	1	1	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
x128	x64	x32	x16	x8	x4	x2	x1
	64	+			4	+	2
<hr/>							
70							

Bahnhofsuhr

Bahnhofsuhr

Wie spät ist es?



Source: bahnhofsuhrsg.ch

Bahnhofsuhr

Wie spät ist es?



Source: bahnhofsuhrsg.ch

07:20:40

Und jetzt andersrum!

Und jetzt andersrum!

Frage

Nun wollen wir andersrum rechnen, also von Dezimalrepräsentation zur Binärrepräsentation. Wie könnten wir den Code aus der vorherigen Aufgabe abändern um die letzten drei bits der Binärrepräsentation zu erhalten?

Lösung

```
#include <iostream>
int main() {
    int a;
    // input
    std::cin >> a;
    // computation
    int digit0 = a % 10;
    int remainder0 = a / 10;
    int digit1 = remainder0 % 10;
    int remainder1 = remainder0 / 10;
    int digit2 = remainder1 % 10;
    // output
    std::cout << digit2 << " " << digit1 << " " << digit0 << std::endl;
    return 0;
}
```

Bedeutung?

Bedeutung?

Frage

Welche Bedeutung haben 10 oder 2 in diesem Programm?

Bedeutung?

$$10^2 \ 10^1 \ 10^0 \\ 517$$

$$7 \cdot 10^0 = 7$$

$$1 \cdot 10^1 = 10$$

$$5 \cdot 10^2 = 500$$

Frage

Welche Bedeutung haben 10 oder 2 in diesem Programm?

$$\underline{\underline{517}}$$

Antwort

Sie sind die Basis in die das Programm umrechnet!

Bedeutung?

Frage

Welche Bedeutung haben 10 oder 2 in diesem Programm?

Antwort

Sie sind die Basis in die das Programm umrechnet!

Und jetzt andersrum!

Frage

Nun wollen wir andersrum rechnen, also von Dezimalrepräsentation zur Binärrepräsentation. Wie könnten wir den Code aus der vorherigen Aufgabe abändern um die letzten drei bits der Binärrepräsentation zu erhalten?

Und jetzt andersrum!

Frage

Nun wollen wir andersrum rechnen, also von Dezimalrepräsentation zur Binärrepräsentation. Wie könnten wir den Code aus der vorherigen Aufgabe abändern um die letzten drei bits der Binärrepräsentation zu erhalten?

Antwort

Einfach alle 10 durch 2 ersetzen!

Lösung

```
#include <iostream>
int main() {
    int a;
    // input
    std::cin >> a;
    // computation
    int digit0 = a % 2;
    int remainder0 = a / 2;
    int digit1 = remainder0 % 2;
    int remainder1 = remainder0 / 2;
    int digit2 = remainder1 % 2;
    // output
    std::cout << digit2 << " " << digit1 << " " << digit0 << std::endl;
    return 0;
}
```

Binärrepräsentation

Frage

Wie sieht ein Algorithmus aus, der die Binärrepräsentation einer Dezimalzahl berechnet?

Wie würde man beispielsweise bei der Dezimalzahl 61_{10} vorgehen, um sie in Binärrepräsentation zu bringen?

Binärrepräsentation

Frage

Wie sieht ein Algorithmus aus, der die Binärrepräsentation einer Dezimalzahl berechnet?

Wie würde man beispielsweise bei der Dezimalzahl 61_{10} vorgehen, um sie in Binärrepräsentation zu bringen?

Antwort

Teile die Dezimaldarstellung durch 2 und behalte den Rest (wie eine Modulodivision). Teile dann die übrige Nummer nochmal und so weiter, bis man 0 erreicht.

Binärrepräsentation

$$61 = 2 * 30 + 1$$

$$30 = 2 * 15 + 0$$

$$15 = 2 * 7 + 1$$

$$7 = 2 * 3 + 1$$

$$3 = 2 * 1 + 1$$

$$1 = 2 * 0 + 1$$

Binärrepräsentation

$$61 = 2 * 30 + 1$$

$$30 = 2 * 15 + 0$$

$$15 = 2 * 7 + 1$$

$$7 = 2 * 3 + 1$$

$$3 = 2 * 1 + 1$$

$$1 = 2 * 0 + 1$$

Dann die letzte Spalte von unten nach oben ablesen und fertig!

$$61_{10} = 111101_2$$

Fragen/Unklarheiten?

6. Expressions und Evaluationen

Was sind Expressions?

Expressions

...sind Ausdrücke die evaluiert werden können. Sie kommen oft in Form von mathematischen Ausdrücken vor, die man dann Stück für Stück evaluiert.

Beispiel “Expression”

Evaluiere die Expression² $5u + 5 * 3u$

²Hier steht `u` für `unsigned int`. Der Wert muss dementsprechen gerechnet werden und es muss auf die Präzedenz der Operatoren (“Punkt vor Strich”) und die Datentypen (später mehr) geachtet werden.

Beispiel “Expression”

Evaluere die Expression² $5u + 5 * 3u$

$5u + 5 * 3u$

²Hier steht `u` für `unsigned int`. Der Wert muss dementsprechen gerechnet werden und es muss auf die Präzedenz der Operatoren (“Punkt vor Strich”) und die Datentypen (später mehr) geachtet werden.

Beispiel “Expression”

Evaluiere die Expression² $5u + 5 * 3u$

$5u + 5 * 3u$ Punkt vor Strich

²Hier steht u für **unsigned int**. Der Wert muss dementsprechen gerechnet werden und es muss auf die Präzedenz der Operatoren (“Punkt vor Strich”) und die Datentypen (später mehr) geachtet werden.

Beispiel “Expression”

Evaluere die Expression² $5u + 5 * 3u$

$5u + 5 * 3u$ Punkt vor Strich

$5u + (5 * 3u)$

²Hier steht u für **unsigned int**. Der Wert muss dementsprechen gerechnet werden und es muss auf die Präzedenz der Operatoren (“Punkt vor Strich”) und die Datentypen (später mehr) geachtet werden.

Beispiel “Expression”

Evaluiere die Expression² `5u + 5 * 3u`

`5u + 5 * 3u` Punkt vor Strich

`5u + (5 * 3u)` das Resultat wird zu `unsigned int`

²Hier steht `u` für `unsigned int`. Der Wert muss dementsprechen gerechnet werden und es muss auf die Präzedenz der Operatoren (“Punkt vor Strich”) und die Datentypen (später mehr) geachtet werden.

Beispiel “Expression”

Evaluiere die Expression² $5u + 5 * 3u$

$5u + 5 * 3u$ Punkt vor Strich

$5u + (5 * 3u)$ das Resultat wird zu **unsigned int**

$5u + 15u$

²Hier steht u für **unsigned int**. Der Wert muss dementsprechen gerechnet werden und es muss auf die Präzedenz der Operatoren (“Punkt vor Strich”) und die Datentypen (später mehr) geachtet werden.

Beispiel “Expression”

Evaluiere die Expression² $5u + 5 * 3u$

$5u + 5 * 3u$ Punkt vor Strich

$5u + (5 * 3u)$ das Resultat wird zu **unsigned int**

$5u + 15u$ simple Addition

²Hier steht u für **unsigned int**. Der Wert muss dementsprechen gerechnet werden und es muss auf die Präzedenz der Operatoren (“Punkt vor Strich”) und die Datentypen (später mehr) geachtet werden.

Beispiel “Expression”

Evaluere die Expression² $5u + 5 * 3u$

$5u + 5 * 3u$ Punkt vor Strich

$5u + (5 * 3u)$ das Resultat wird zu **unsigned int**

$5u + 15u$ simple Addition

$20u$

²Hier steht u für **unsigned int**. Der Wert muss dementsprechen gerechnet werden und es muss auf die Präzedenz der Operatoren (“Punkt vor Strich”) und die Datentypen (später mehr) geachtet werden.

Valide Expression erkennen

Welche sind eine valide C++-Expression und welche nicht?

1. $1*(2*3)$

2. $(a=1)$

3. $(1$

4. $(a*3) = (b*5)$

Valide Expression erkennen

Welche sind eine valide C++-Expression und welche nicht?

1. $1*(2*3)$

2. $(a=1)$

Lösungen

$(a, 1)$
↓
 a ✓

$\boxed{1}$
 a

3. $(1$ $a=2;$
4. $(a*3) = (b*5)$ $b=1;$
 (6) ← (5)

Valide Expression erkennen

Welche sind eine valide C++-Expression und welche nicht?

1. $1*(2*3)$

2. $(a=1)$

3. $(1$

4. $(a*3) = (b*5)$

Lösungen

- 3. ist nicht valide, da die Klammer () nicht geschlossen wird

Valide Expression erkennen

Welche sind eine valide C++-Expression und welche nicht?

1. $1*(2*3)$

2. $(a=1)$

3. $(1$

4. $(a*3) = (b*5)$

Lösungen

- 3. ist nicht valide, da die Klammer () nicht geschlossen wird
- 4. ist nicht valide, da $(a*3)$ zu einem R-value wird, der Assignment-Operator (=) aber ein L-value zu seiner Linken erwartet

Valide Expression erkennen

Welche sind eine valide C++-Expression und welche nicht?

1. $1 * (2 * 3)$
2. $(a - 1)$

3. $(1$

4. $(a * 3) = (b * 5)$

Lösungen

- 3. ist nicht valide, da die Klammer () nicht geschlossen wird
- 4. ist nicht valide, da $(a * 3)$ zu einem R-value wird, der Assignment-Operator (=) aber ein L-value zu seiner Linken erwartet
- 1. und 2. sind valide C++-Expressions

Valide Expression erkennen

Welche sind L-Values und welche sind R-Values?

1. $1*(2*3)$

2. $(a=1)$

3. $(1$

4. $(a*3) = (b*5)$

Valide Expression erkennen

Welche sind L-Values und welche sind R-Values?

1. $1*(2*3)$

2. $(a=1)$

3. $(1$

4. $(a*3) = (b*5)$

Lösungen

Valide Expression erkennen

Welche sind L-Values und welche sind R-Values?

1. $1*(2*3)$

2. $(a=1)$

3. $(1$

4. $(a*3) = (b*5)$

Lösungen

- 3. und 4. sind weder noch, da sie keine validen Expressions sind

Valide Expression erkennen

Welche sind L-Values und welche sind R-Values?

1. $1*(2*3)$

2. $(a=1)$

3. $(1$

4. $(a*3) = (b*5)$

Lösungen

- 3. und 4. sind weder noch, da sie keine validen Expressions sind
- 1. ist ein R-Value, da der Multiplikation-Operator (*) ein R-Value zurückgibt

Valide Expression erkennen

Welche sind L-Values und welche sind R-Values?

1. $1*(2*3)$

2. $(a=1)$

3. $(1$

4. $(a*3) = (b*5)$

Lösungen

- 3. und 4. sind weder noch, da sie keine validen Expressions sind
- 1. ist ein R-Value, da der Multiplikation-Operator ($*$) ein R-Value zurückgibt
- 2. ist ein L-Value, da der Assignment-Operator ($=$) ein L-Value zurückgibt

Valide Expression erkennen

Zu was werden die Expressions evaluiert?

1. $1*(2*3)$

2. $(a=1)$

3. $(1$

4. $(a*3) = (b*5)$

Valide Expression erkennen

Zu was werden die Expressions evaluiert?

1. $1*(2*3)$

2. $(a=1)$

3. $(1$

4. $(a*3) = (b*5)$

Lösungen

Valide Expression erkennen

Zu was werden die Expressions evaluiert?

1. $1*(2*3)$

2. $(a=1)$

3. $(1$

4. $(a*3) = (b*5)$

Lösungen

- 3. und 4. können nicht evaluiert werden, da sie keine validen Expressions sind

Valide Expression erkennen

Zu was werden die Expressions evaluiert?

1. $1*(2*3)$

2. $(a=1)$

3. $(1$

4. $(a*3) = (b*5)$

Lösungen

- 3. und 4. können nicht evaluiert werden, da sie keine validen Expressions sind
- 1. evaluiert zu 6, also das Ergebnis der simplen Multiplikationen

Valide Expression erkennen

Zu was werden die Expressions evaluiert?

1. $1*(2*3)$

2. $(a=1)$

3. $(1$

4. $(a*3) = (b*5)$

Lösungen

- 3. und 4. können nicht evaluiert werden, da sie keine validen Expressions sind
- 1. evaluiert zu 6, also das Ergebnis der simplen Multiplikationen
- 2. evaluiert zu 1, da a zurückgegeben wird, dem gerade davor der Wert 1 zugewiesen wurde

Fragen/Unklarheiten?

7. Binärrepräsentation von neg. Integern

Binärrepräsentation von negativen Integern

Aufgabe

Finde einen Weg, um negative Integer abzuspeichern.

Tipp:

$$n + (-n) = 0$$

Binärrepräsentation von negativen Integern

Aufgabe

Finde einen Weg, um negative Integer abzuspeichern.

Tipp:

$$n + (-n) = 0$$

Lösung

Behandle die vorderste Ziffer als das negative ihres “eigentlichen” Werts

 Sehr gutes Video dazu

Binärrepräsentation von negativen Integern

Aufgabe

Wie erhält man die (*signed*) `int`-Repräsentation einer beliebigen, negativen Ganzzahl $n < 0$?

Binärrepräsentation von negativen Integern

Aufgabe

Wie erhält man die (*signed*) `int`-Repräsentation einer beliebigen, negativen Ganzzahl $n < 0$?

Lösung

1. Absolutbetrag von n bestimmen
2. Absolutbetrag von n in Binärrepräsentation aufschreiben
3. Bits flippen
4. 1 addieren

Fragen/Unklarheiten?

8. Tipps zu **code** expert

Tipps für **code expert**

- Früh genug anfangen
- Zusammen daran arbeiten (aber nicht abschreiben!)

9. Outro

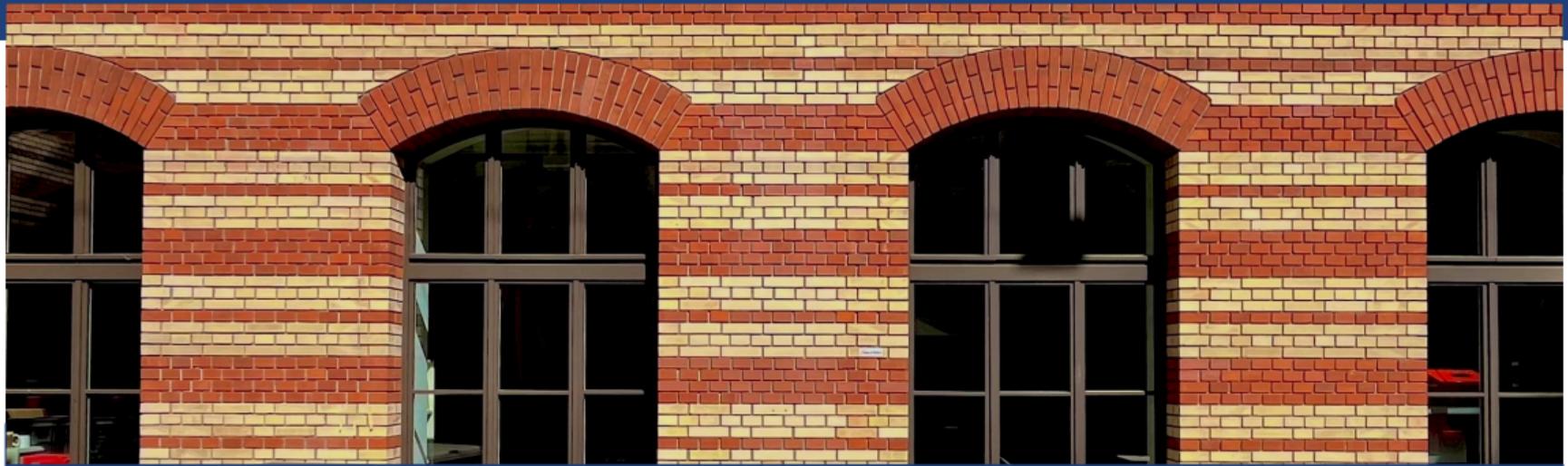
Allgemeine Fragen?

Bis zum nächsten Mal

Next week:

- further code export - intro
- LaTeX + mark down!

Schöne Woche!



Übungsstunde — Informatik — 02

Adel Gavranović

Boolean Expressions, `for`-Schleifen, Debugging, Selection Statements

Heutige Themen

Boolean Expressions

for-Schleifen

Debugging

Debugging in **code expert**

Selection Statements



`n.ethz.ch/~agavranovic`

 Material

 Webpage

 E-Mail

1. Lernziele

Ziele für Heute

Lernziele

- Verstehen, was ein *Short Circuit* ist
- Verstehen, was Precedence und Assoziativität im Kontext von Bool'schen Operatoren sind
- (Bool'sche) Expressions von Hand Evaluieren können
- Expressions vereinfachen können
- Program Tracing* verstehen und anwenden können
- for**-Schleifen verstehen und anwenden können
- Simple, "manuelle" Debugging-Strategien anwenden können

2. Zusammenfassung

Fragen/Unklarheiten?

3. Boolean Expressions

Booleans

Booleans

`bool` ist ein Datentyp der nur zwei Werte annehmen kann:

Booleans

`bool` ist ein Datentyp der nur zwei Werte annehmen kann: `{true, false}`

Booleans

`bool` ist ein Datentyp der nur zwei Werte annehmen kann: `{true, false}`

Konvertierung "casting"

`(bool) 5`
↑
`bool`

Booleans

`bool` ist ein Datentyp der nur zwei Werte annehmen kann: `{true, false}`

Konvertierung

- Bool zu Zahl

Booleans

`bool` ist ein Datentyp der nur zwei Werte annehmen kann: `{true, false}`

Konvertierung

- Bool zu Zahl
 - wenn `true` in eine Zahl umgewandelt wird,

Booleans

`bool` ist ein Datentyp der nur zwei Werte annehmen kann: `{true, false}`

Konvertierung

- Bool zu Zahl
 - wenn `true` in eine Zahl umgewandelt wird, wird es die Zahl 1 sein

Booleans

`bool` ist ein Datentyp der nur zwei Werte annehmen kann: `{true, false}`

Konvertierung

- Bool zu Zahl
 - wenn `true` in eine Zahl umgewandelt wird, wird es die Zahl 1 sein
 - wenn `false` in eine Zahl umgewandelt wird,

Booleans

`bool` ist ein Datentyp der nur zwei Werte annehmen kann: `{true, false}`

Konvertierung

- Bool zu Zahl
 - wenn `true` in eine Zahl umgewandelt wird, wird es die Zahl 1 sein
 - wenn `false` in eine Zahl umgewandelt wird, wird es die Zahl 0 sein

Booleans

`bool` ist ein Datentyp der nur zwei Werte annehmen kann: `{true, false}`

Konvertierung

- Bool zu Zahl
 - wenn `true` in eine Zahl umgewandelt wird, wird es die Zahl 1 sein
 - wenn `false` in eine Zahl umgewandelt wird, wird es die Zahl 0 sein
- Zahl zu Bool

Booleans

`bool` ist ein Datentyp der nur zwei Werte annehmen kann: `{true, false}`

Konvertierung

- Bool zu Zahl
 - wenn `true` in eine Zahl umgewandelt wird, wird es die Zahl 1 sein
 - wenn `false` in eine Zahl umgewandelt wird, wird es die Zahl 0 sein
- Zahl zu Bool
 - wenn 0 in einen `bool` umgewandelt wird,

Booleans

`bool` ist ein Datentyp der nur zwei Werte annehmen kann: `{true, false}`

Konvertierung

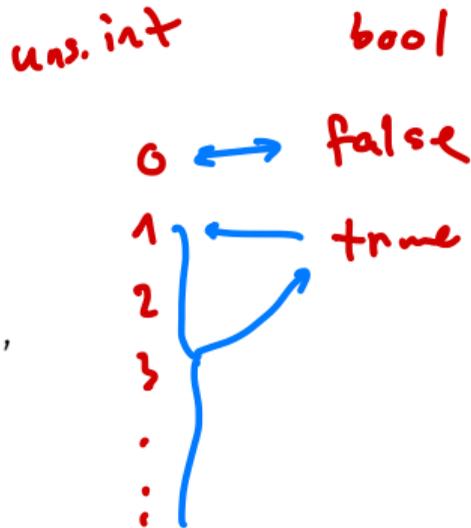
- Bool zu Zahl
 - wenn `true` in eine Zahl umgewandelt wird, wird es die Zahl 1 sein
 - wenn `false` in eine Zahl umgewandelt wird, wird es die Zahl 0 sein
- Zahl zu Bool
 - wenn 0 in einen `bool` umgewandelt wird, wird es der Wert `false` sein
 - wenn eine Zahl $\neq 0$ in einen `bool` umgewandelt wird,

Booleans

`bool` ist ein Datentyp der nur zwei Werte annehmen kann: `{true, false}`

Konvertierung

- Bool zu Zahl
 - wenn `true` in eine Zahl umgewandelt wird, wird es die Zahl 1 sein
 - wenn `false` in eine Zahl umgewandelt wird, wird es die Zahl 0 sein
- Zahl zu Bool
 - wenn 0 in einen `bool` umgewandelt wird, wird es der Wert `false` sein
 - wenn eine Zahl $\neq 0$ in einen `bool` umgewandelt wird, wird es der Wert `true` sein



Precedence Ranking

Precedence Ranking¹

¹Gezeigt sind hier nur die z. Z. wichtigsten. Für den Rest, siehe [cppreference](#)

Precedence Ranking

Precedence Ranking¹

1. a++, a--
2. ++a, --a, -a, !a, *a, &a
3. *, /, %
4. +, -
5. <, <=, >, >=
6. ==, !=
7. &&
8. ||
9. =, +=, -=, *=, /=, %=

(noch nicht relevant)

```
a = a + 5;  
a += 5;
```

¹Gezeigt sind hier nur die z. Z. wichtigsten. Für den Rest, siehe [cppreference](#)

(Nutzt) (Klammern)

- Klammern funktionieren wie in der Mathematik
- sie werden oft gebraucht, um die korrekte Evaluation offensichtlich zu machen
- ...oder eben um die Evaluationsreihenfolge abzuändern

(Nutzt) (Klammern)

- Klammern funktionieren wie in der Mathematik
- sie werden oft gebraucht, um die korrekte Evaluation offensichtlich zu machen
- ...oder eben um die Evaluationsreihenfolge abzuändern

Aufgabe

Mache die Evaluation mittels Klammern offensichtlich:

$(3 < (4 + 1)) \&\& (2 < 3)$

Tipp: nutze die vorherige Folie

(Nutzt) (Klammern)

- Klammern funktionieren wie in der Mathematik
- sie werden oft gebraucht, um die korrekte Evaluation offensichtlich zu machen
- ...oder eben um die Evaluationsreihenfolge abzuändern

Aufgabe

Mache die Evaluation mittels Klammern offensichtlich:

$3 < 4 + 1 \ \&\& \ 2 < 3$

Tipp: nutze die vorherige Folie

Lösung

$(3 < (4 + 1)) \ \&\& \ (2 < 3)$

\equiv

Mehrere Operatoren mit gleicher Präzedenz

Assoziativität

Mehrere Operatoren mit gleicher Präzedenz

Assoziativität

- Sagt aus, in welcher Reihenfolge Operationen evaluiert werden

Mehrere Operatoren mit gleicher Präzedenz

Assoziativität

- Sagt aus, in welcher Reihenfolge Operationen evaluiert werden
- Entweder Right-to-left (\leftarrow) oder Left-to-right (\rightarrow)

$$a = (b = (c = (d = e))) ;$$

Mehrere Operatoren mit gleicher Präzedenz

Assoziativität

- Sagt aus, in welcher Reihenfolge Operationen evaluiert werden
- Entweder Right-to-left (\leftarrow) oder Left-to-right (\rightarrow)
- 💡 Am besten eine Tabelle auf die Zusammenfassung mit Präzedenzen und Assoziativitäten schreiben

Mehrere Operatoren mit gleicher Präzedenz

Kurzaufgabe

Wie würdest du klammern, damit die Evaluation der unteren Expression offensichtlich wird?

Tipp: Die Assoziativität von `&&` ist Left-to-right

`(a && b) && c`

Mehrere Operatoren mit gleicher Präzedenz

Kurzaufgabe

Wie würdest du klammern, damit die Evaluation der unteren Expression offensichtlich wird?

Tipp: Die Assoziativität von `&&` ist Left-to-right

`a && b && c`

Lösung

`(a && b) && c`

Short Circuits (deutsch: Kurzschluss)

Die bool'schen Operatoren `&&` und `||` evaluieren zuerst die linke Expression und dann *nur falls nötig* die rechte Expression.

`false && (---)`
false

`(true) || (---)`
true

Short Circuits (deutsch: Kurzschluss)

Die bool'schen Operatoren `&&` und `||` evaluieren zuerst die linke Expression und dann *nur falls nötig* die rechte Expression.

Insbesondere heisst das, dass die rechte Expression *nicht* evaluiert wird, wenn man das Ergebnis der gesamten Evaluation bereits herleiten kann.

Short Circuits in Code

true

```
if(3 > 2 && (10 > 11)){  
    std::cout << "Of course not!\n";  
} // not a short circuit evaluation
```

Short Circuits in Code

```
int a = 3;
```

```
if (false && ++a < 2) {
```

```
    std::cout << "Of course not!\n";
```

```
} // a short circuit evaluation
```

```
std::cout << a << "\n"; // what will be the output? 3
```

```
if (++a < 2 && false) {
```

```
    std::cout << "Of course not!\n";
```

```
} // another short circuit evaluation
```

```
std::cout << a << "\n"; // what will be the output? 4
```

false

never happened!

happens!

Verständniskontrolle I

Aufgabe

Evaluieren Sie die folgende Expression von Hand und notieren Sie jeden Zwischenschritt. Nehmen an `int x = 1`, `int y = 1`:

`2 > 3 && 17 * u - 55 <= ++x + y`
17u

Verständniskontrolle I

Aufgabe

Evaluieren Sie die folgende Expression von Hand und notieren Sie jeden Zwischenschritt. Nehmen Sie an `int x = 1`, `int y = 1`:

`2 > 3 && 17 * 55 <= ++x + y`

Lösung

`(2 > 3) && 17 * 55 <= ++x + y` beginne links
false

Verständniskontrolle I

Aufgabe

Evaluire die folgende Expression von Hand und notiere jeden Zwischenschritt. Nehme an `int x = 1`, `int y = 1`:

```
2>3 && 17 u - 55 <= ++x + y
```

Lösung

```
(2>3) && 17 u - 55 <= ++x + y  beginne links  
false && 17 u - 55 <= ++x + y  short circuit!
```

Verständniskontrolle I

Aufgabe

Evaluieren Sie die folgende Expression von Hand und notieren Sie jeden Zwischenschritt. Nehmen Sie an `int x = 1`, `int y = 1`:

```
2 > 3 && 17 * 55 <= ++x + y
```

Lösung

```
(2 > 3) && 17 * 55 <= ++x + y
```

 beginne links

```
false && 17 * 55 <= ++x + y
```

 short circuit!

```
false
```

Merke: `(true || ...)` evaluiert immer zu `true`

Verständniskontrolle II

Aufgabe

Evaluere die folgende Expression von Hand und notiere jeden Zwischenschritt. Nehme an `int x = 1`:

`!(1 < 2 && x == 1) + 1`

Verständniskontrolle II

Aufgabe

Evaluere die folgende Expression von Hand und notiere jeden Zwischenschritt. Nehme an `int x = 1`:

```
!(1 < 2 && x == 1) + 1
```

Lösung

```
!(1 < 2 && x == 1) + 1
```

Verständniskontrolle II

Aufgabe

Evaluere die folgende Expression von Hand und notiere jeden Zwischenschritt. Nehme an `int x = 1`:

```
!(1 < 2 && x == 1) + 1
```

Lösung

```
!(1 < 2 && x == 1) + 1
```

```
(!((1 < 2) && (x == 1))) + 1
```

Verständniskontrolle II

Aufgabe

Evaluieren Sie die folgende Expression von Hand und notieren Sie jeden Zwischenschritt. Nehmen Sie an `int x = 1`:

```
!(1 < 2 && x == 1) + 1
```

Lösung

```
!(1 < 2 && x == 1) + 1
```

```
(!((1 < 2) && (x == 1))) + 1
```

```
(!(true) && (true))) + 1
```

Verständniskontrolle II

Aufgabe

Evaluere die folgende Expression von Hand und notiere jeden Zwischenschritt. Nehme an `int x = 1`:

```
!(1 < 2 && x == 1) + 1
```

Lösung

```
!(1 < 2 && x == 1) + 1
```

```
(!((1 < 2) && (x == 1))) + 1
```

```
(!(true) && (true))) + 1
```

```
!(true) + 1
```

Verständniskontrolle II

Aufgabe

Evaluieren Sie die folgende Expression von Hand und notieren Sie jeden Zwischenschritt. Nehmen Sie an `int x = 1`:

```
!(1 < 2 && x == 1) + 1
```

Lösung

```
!(1 < 2 && x == 1) + 1
```

```
(!((1 < 2) && (x == 1))) + 1
```

```
(!(true) && (true)) + 1
```

```
!(true) + 1
```

```
false + 1
```

Verständniskontrolle II

Aufgabe

Evaluieren Sie die folgende Expression von Hand und notieren Sie jeden Zwischenschritt. Nehmen Sie an `int x = 1`:

```
!(1 < 2 && x == 1) + 1
```

Lösung

```
!(1 < 2 && x == 1) + 1
```

```
(!((1 < 2) && (x == 1))) + 1
```

```
(!(true) && true) + 1
```

```
!(true) + 1
```

```
false + 1
```

```
0 + 1
```

Verständniskontrolle II

Aufgabe

Evaluere die folgende Expression von Hand und notiere jeden Zwischenschritt. Nehme an `int x = 1`:

```
!(1 < 2 && x == 1) + 1
```

Lösung

```
!(1 < 2 && x == 1) + 1
```

```
(!((1 < 2) && (x == 1))) + 1
```

```
(!(true) && (true))) + 1
```

```
!(true) + 1
```

```
false + 1
```

```
0 + 1
```

```
1
```

Verständniskontrolle III

Aufgabe

Evaluieren Sie die folgende Expression von Hand und notieren Sie jeden Zwischenschritt. Nehmen Sie an `int x = 1`:

`x == 1 || 1 / (x - 1) < 1`

Verständniskontrolle III

Aufgabe

Evaluieren Sie die folgende Expression von Hand und notieren Sie jeden Zwischenschritt. Nehmen Sie an `int x = 1`:

`x == 1 || 1 / (x - 1) < 1`

Lösung

Zuerst: Klammern!

`(x == 1) || ((1 / (x - 1)) < 1)` beginne links

Verständniskontrolle III

Aufgabe

Evaluieren Sie die folgende Expression von Hand und notieren Sie jeden Zwischenschritt. Nehmen Sie an `int x = 1`:

```
x == 1 || 1 / (x - 1) < 1
```

Lösung

Zuerst: Klammern!

```
(x == 1) || ((1 / (x - 1)) < 1)    beginne links
```

```
(1 == 1) || ((1 / (x - 1)) < 1)
```

Verständniskontrolle III

Aufgabe

Evaluieren Sie die folgende Expression von Hand und notieren Sie jeden Zwischenschritt. Nehmen Sie an `int x = 1`:

```
x == 1 || 1 / (x - 1) < 1
```

Lösung

Zuerst: Klammern!

```
(x == 1) || ((1 / (x - 1)) < 1)    beginne links
```

```
(1 == 1) || ((1 / (x - 1)) < 1)
```

```
true || ((1 / (x - 1)) < 1)    short circuit!
```

Verständniskontrolle III

Aufgabe

Evaluieren Sie die folgende Expression von Hand und notieren Sie jeden Zwischenschritt. Nehmen Sie an `int x = 1`:

```
x == 1 || 1 / (x - 1) < 1
```

Lösung

Zuerst: Klammern!

```
(x == 1) || ((1 / (x - 1)) < 1)    beginne links
```

```
(1 == 1) || ((1 / (x - 1)) < 1)
```

```
true || ((1 / (x - 1)) < 1)    short circuit!
```

```
true
```

Fragen/Unklarheiten?

Minimale Expression

Oft kann man Ausdrücke vereinfachen, um sie leichter zu verstehen oder um sie eleganter zu gestalten und weniger Aufmerksamkeit zu erregen.
Beispielsweise:

```
(x > 3) == true || z - 3 >= 0
```

kann vereinfacht werden zu

Minimale Expression

Oft kann man Ausdrücke vereinfachen, um sie leichter zu verstehen oder um sie eleganter zu gestalten und weniger Aufmerksamkeit zu erregen.
Beispielsweise:

```
(x > 3) == true || z - 3 >= 0
```

kann vereinfacht werden zu

```
x > 3 || z >= 3
```

Verständniskontrolle I

Aufgabe

Minimiere die die folgenden booleschen Expressions. Nehme an

`bool a,b; int n;`:

`b == true && !(n < 0) && (n != 0) && (n != 6) && (n < 6)`

Verständniskontrolle I

Aufgabe

Minimiere die die folgenden booleschen Expressions. Nehme an

```
bool a,b; int n;:
```

```
b == true && !(n < 0) && (n != 0) && (n != 6) && (n < 6)
```

Lösung

```
b && n > 0 && n < 6
```

Verständniskontrolle I

Aufgabe

Minimiere die die folgenden booleschen Expressions. Nehme an

```
bool a,b; int n;:
```

```
b == true && !(n < 0) && (n != 0) && (n != 6) && (n < 6)
```

Lösung

```
b && n > 0 && n < 6
```

oder um es noch deutlicher zu machen:

```
b && (n > 0) && (n < 6)
```

Verständniskontrolle I

Aufgabe

Minimiere die die folgenden booleschen Expressions. Nehme an

`bool a,b; int n;`

`b == true && !(n < 0) && (n != 0) && (n != 6) && (n < 6)`

Lösung

`b && n > 0 && n < 6`

oder um es noch deutlicher zu machen:

`b && (n > 0) && (n < 6)`

Das ist sehr falsch:

`b && (0 < n < 6)`

Verständniskontrolle I

Aufgabe

Minimiere die die folgenden booleschen Expressions. Nehme an

`bool a,b; int n;`

`b == true && !(n < 0) && (n != 0) && (n != 6) && (n < 6)`

Lösung

`b && n > 0 && n < 6`

oder um es noch deutlicher zu machen:

`b && (n > 0) && (n < 6)`

Das ist sehr falsch:

`b && (0 < n < 6)` Aber wieso?

Verständniskontrolle II

Aufgabe

Minimiere die die folgenden booleschen Expressions. Nehme an

```
bool a,b; int n;
```

```
a && !(a == true) || (b == false)
```

Verständniskontrolle II

Aufgabe

Minimiere die die folgenden booleschen Expressions. Nehme an

```
bool a,b; int n;
```

```
a && !(a == true) || (b == false)
```

Lösung

```
!b
```

Verständniskontrolle III

Aufgabe

Minimiere die die folgenden booleschen Expressions. Nehme an

`bool a,b; int n;`

```
!(a != false && !(b == false) == false)
```



Verständniskontrolle III

Aufgabe

Minimiere die die folgenden booleschen Expressions. Nehme an

```
bool a,b; int n;
```

```
!(a != false && !(b == false) == false)
```

Lösung

```
!a || b
```

Fragen/Unklarheiten?

4. for-Schleifen

Kurzeinführung zu Scopes

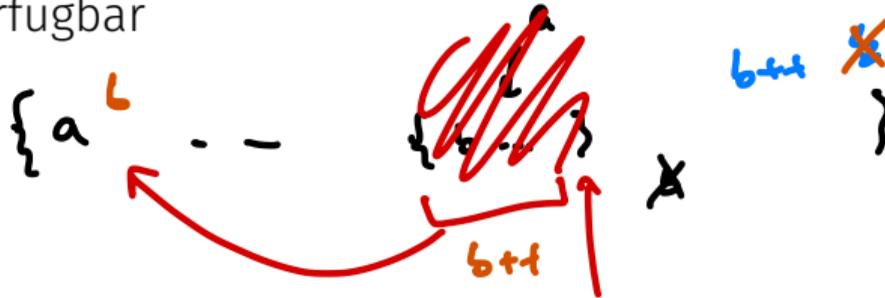
- Praktisch immer, wenn Ihr {diese geschweiften Klammern} verwendet, erstellt ihr einen {Scope}

Kurzeinführung zu Scopes

- Praktisch immer, wenn Ihr {diese geschweiften Klammern} verwendet, erstellt ihr einen {Scope}
- Stellt euch einen Scope als eine "Welt in einer Welt" vor

Kurzeinführung zu Scopes

- Praktisch immer, wenn Ihr {diese geschweiften Klammern} verwendet, erstellt ihr einen {Scope}
- Stellt euch einen Scope als eine "Welt in einer Welt" vor
- Informationen/Variablen können nicht aus einem Scope rausfließen, aber Informationen/Variablen von aussen sind im inneren Scope verfügbar



Kurzeinführung zu Scopes

- Praktisch immer, wenn Ihr {diese geschweiften Klammern} verwendet, erstellt ihr einen {Scope}
- Stellt euch einen Scope als eine "Welt in einer Welt" vor
- Informationen/Variablen können nicht aus einem Scope rausfließen, aber Informationen/Variablen von aussen sind im inneren Scope verfügbar
- Wenn das Programm bei der rechten geschweiften Klammer des Scopes ankommt, stirbt jegliche Information/Variable innerhalb dieses Scopes

Allgemeine Struktur einer for-Schleife

```
for(①init; ②condition; ③expression){  
    statement 1;  
    statement 2; ③  
    // ....  
}
```

Wichtige Bemerkung

Der `expression`-Teil wird *nach* den statements ausgeführt.

Program Tracing

"*Program Tracing* ist der Prozess des Ausführens eines Programms von Hand mit konkreten Eingaben."

Ziemlich wichtiger Skill, insbesondere am Anfang. Irgendwann werdet ihr das quasi im Kopf können. Einen ausführlichen Guide dazu findet ihr hier:

 [Program Tracing](#)

for-Schleifen-Beispiel

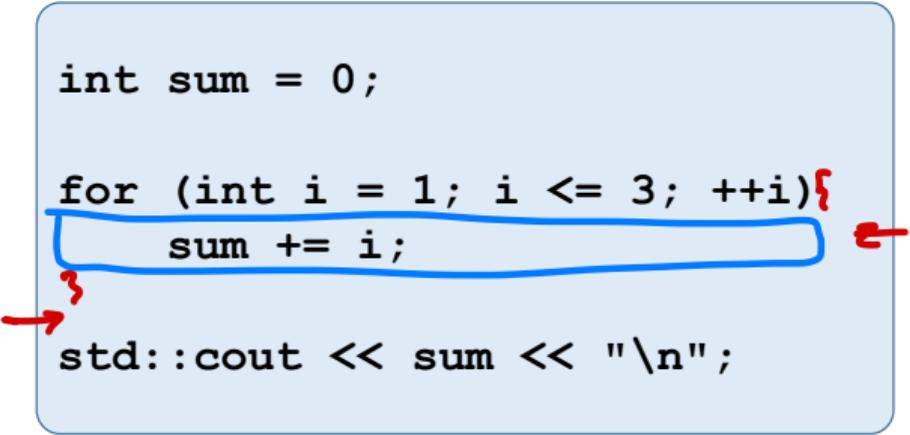
for Loop

Example – for Loop

```
int sum = 0;

for (int i = 1; i <= 3; ++i){
    sum += i;
}

std::cout << sum << "\n";
```



Example – for Loop

sum: 0

```
int sum = 0;  
  
for (int i = 1; i <= 3; ++i)  
    sum += i;  
  
std::cout << sum << "\n";
```

Example – for Loop

```
int sum = 0;

for (int i = 1; i <= 3; ++i)
    sum += i;

std::cout << sum << "\n";
```

sum:	0
i:	1

Example – for Loop

sum:	0
i:	1

```
int sum = 0;

for (int i = 1; i <= 3; ++i)
    sum += i;

std::cout << sum << "\n";
```

Example – for Loop

```
int sum = 0;
```

```
for (int i = 1; i <= 3; ++i)  
    sum += i;
```

```
std::cout << sum << "\n";
```

1 <= 3

true

sum: 0

i: 1

Example – for Loop

```
int sum = 0;

for (int i = 1; i <= 3; ++i)
    sum += i;

std::cout << sum << "\n";
```

sum:	1
i:	1

Example – for Loop

```
int sum = 0;

for (int i = 1; i <= 3; ++i)
    sum += i;

std::cout << sum << "\n";
```

sum:	1
i:	2



Example – for Loop

sum:	1
i:	2

```
int sum = 0;

for (int i = 1; i <= 3; ++i)
    sum += i;

std::cout << sum << "\n";
```

Example – for Loop

```
int sum = 0;
```

```
for (int i = 1; i <= 3; ++i)  
    sum += i;
```

```
std::cout << sum << "\n";
```

2 <= 3

true

sum:	1
i:	2

Example – for Loop

```
int sum = 0;

for (int i = 1; i <= 3; ++i)
    sum += i;

std::cout << sum << "\n";
```

sum:	3
i:	2

Example – for Loop

```
int sum = 0;

for (int i = 1; i <= 3; ++i)
    sum += i;

std::cout << sum << "\n";
```

sum:	3
i:	3

Example – for Loop

```
int sum = 0;

for (int i = 1; i <= 3; ++i)
    sum += i;

std::cout << sum << "\n";
```

sum:	3
i:	3

Example – for Loop

```
int sum = 0;
```

```
for (int i = 1; i <= 3; ++i)  
    sum += i;
```

```
std::cout << sum << "\n";
```

3 <= 3

true

sum: 3

i: 3

Example – for Loop

```
int sum = 0;

for (int i = 1; i <= 3; ++i)
    sum += i;

std::cout << sum << "\n";
```

sum:	6
i:	3

Example – for Loop

```
int sum = 0;

for (int i = 1; i <= 3; ++i)
    sum += i;

std::cout << sum << "\n";
```

sum:	6
i:	4

Example – for Loop

```
int sum = 0;

for (int i = 1; i <= 3; ++i)
    sum += i;

std::cout << sum << "\n";
```

sum:	6
i:	4

Example – for Loop

```
int sum = 0;
```

```
for (int i = 1; i <= 3; ++i)  
    sum += i;
```

```
std::cout << sum << "\n";
```

4 <= 3

false

sum: 6

i: 4

Example – for Loop

sum: 6

```
int sum = 0;

for (int i = 1; i <= 3; ++i){
    sum += i;
}
std::cout << sum << "\n";
```

Fragen/Unklarheiten?

Aufgabe Strange Sum

Aufgabe

Öffnet *Strange Sum* auf [code expert](#) und versucht es zuerst mit Stift und Papier zu lösen. (10 min) ~~10 min~~ ~ 5 min

Description

Write a program that reads a number $n > 0$ from standard input and outputs the sum of all positive numbers up to n that are odd but not divisible by 5.

n
 i :

```
std::cin >> n;
```

i
 $i \cdot 2 == 0$ // even

$i \cdot 2$

$i \cdot 5$

Aufgabe *Strange Sum*

Aufgabe

Öffnet *Strange Sum* auf **code expert** und versucht es zuerst mit Stift und Papier zu lösen. (10min)

Description

Write a program that reads a number $n > 0$ from standard input and outputs the sum of all positive numbers up to n that are odd but not divisible by 5.

Aufgabe Und jetzt schreibt das dazugehörige Programm. (5min)

Fragen/Unklarheiten?

Mögliche Lösung zu *Strange Sum*

```
// input
unsigned int strangesum = 0;
unsigned int n;
std::cin >> n;

// computation
for(unsigned int i = 1; i <= n; i++){
    if((i % 2) == 1){
        if(i % 5){
            strangesum += i;
        }
    }
}

// output
std::cout << strangesum << "\n";
```

Kompaktere Lösung zu *Strange Sum*

```
// input
unsigned int strangesum = 0;
unsigned int n;
std::cin >> n;

// computation
for(unsigned int i = 1; i <= n; i++){
    if( ((i % 2) != 1) && (i % 5) ){
        strangesum += i;
    }
}

// output
std::cout << strangesum << "\n";
```

Noch kompaktere Lösung zu *Strange Sum*

```
// input
unsigned int strangesum = 0;
unsigned int n;
std::cin >> n;

// computation
for(unsigned int i = 1; i <= n; i+=2){
    if(i % 5){
        strangesum += i;
    }
}

// output
std::cout << strangesum << "\n";
```

Aufgabe *Largest Power*

Aufgabe

Öffnet *Largest Power* auf **code expert** und versucht es zuerst mit Stift und Papier zu lösen. (10min)

Description

Write a program that inputs a positive natural number n and outputs the largest number p that is a power of 2 and smaller or equal to n .

Aufgabe *Largest Power*

Aufgabe

Öffnet *Largest Power* auf **code expert** und versucht es zuerst mit Stift und Papier zu lösen. (10min)

Description

Write a program that inputs a positive natural number n and outputs the largest number p that is a power of 2 and smaller or equal to n .

Aufgabe

Und jetzt schreibt das dazugehörige Programm. (5min)

Aufgabe *Largest Power*

Aufgabe

Öffnet *Largest Power* auf **code expert** und versucht es zuerst mit Stift und Papier zu lösen. (10min)

Description

Write a program that inputs a positive natural number n and outputs the largest number p that is a power of 2 and smaller or equal to n .

Aufgabe

Und jetzt schreibt das dazugehörige Programm. (5min)

Aufgabe

Besprecht eure Ansätze mit den Personen neben euch. Hattet ihr den gleichen Ansatz? Was könnt ihr voneinander lernen? (7min)

Mögliche Lösung zu *Largest Power*

```
#include <iostream>
#include <cassert>

int main () {
    unsigned int n;
    std::cin >> n;
    assert(n >= 1);

    unsigned int power = 1;
    for (; power <= n / 2; power *= 2); ←

    std::cout << power << std::endl;

    return 0;
}
```

Fragen/Unklarheiten?

5. Debugging

Debugging

...ist der Prozess des Auffindens und der Behebung von Fehlern (Defekte oder Probleme, die den korrekten Betrieb verhindern) in Programmen, Software oder Systemen.

Debugging

```
int main () {
    const int n = 6;

    // Compute n^12
    int prod = 1;
    for (int i = 1; 1 <= i < 13; ++i) {
        prod *= n;
    }

    // Output stars
    for (int i = 1; i < prod; ++i) {
        std::cout << "*";
    }
    std::cout << "\n";
    return 0;
}
```

Debugging - Live Demo

Frage

Wie könnten wir herausfinden, bei welcher Zeile das Programm feststeckt?

Debugging - Live Demo

Frage

Wie könnten wir herausfinden, bei welcher Zeile das Programm feststeckt?

Antwort

Sachen ausgeben lassen an interessanten Stellen im Code

Debugging - Live Demo

Frage

Wie könnten wir herausfinden, bei welcher Zeile das Programm feststeckt?

Antwort

Sachen ausgeben lassen an interessanten Stellen im Code

Frage

Wieso steckt das Programm in der ersten for-Schleife fest?

Debugging - Live Demo

Frage

Wie könnten wir herausfinden, bei welcher Zeile das Programm feststeckt?

Antwort

Sachen ausgeben lassen an interessanten Stellen im Code

Frage

Wieso steckt das Programm in der ersten for-Schleife fest?

Antwort

Die condition ist falsch geschrieben!

Debugging - Live Demo

Frage

Wie könnten wir herausfinden, bei welcher Zeile das Programm feststeckt?

Antwort

Sachen ausgeben lassen an interessanten Stellen im Code

Frage

Wieso steckt das Programm in der ersten for-Schleife fest?

Antwort

Die condition ist falsch geschrieben!

Sollte sein: `1 <= i && i < 13`.

Debugging - Live Demo

Frage

Wie können wir ermitteln, wieso trotzdem nichts ausgegeben wird?

Debugging - Live Demo

Frage

Wie können wir ermitteln, wieso trotzdem nichts ausgegeben wird?

Antwort

Einfach den Wert von `prod` nach der ersten Schleife ausgeben lassen

Debugging - Live Demo

Frage

Wie können wir ermitteln, wieso trotzdem nichts ausgegeben wird?

Antwort

Einfach den Wert von `prod` nach der ersten Schleife ausgeben lassen

Frage

Wie finden wir raus, wieso `prod` negativ wurde?

Debugging - Live Demo

Frage

Wie können wir ermitteln, wieso trotzdem nichts ausgegeben wird?

Antwort

Einfach den Wert von `prod` nach der ersten Schleife ausgeben lassen

Frage

Wie finden wir raus, wieso `prod` negativ wurde?

Antwort

Einfach den Wert von `prod` in *jeder* Iteration in der Schleife ausgeben lassen

Fragen/Unklarheiten?

Debugging in **code** expert

Debugging in `code expert`

```
int main() {  
  
    long start = 2;  
    // Finding the largest power of 2 representable by long  
    while (true) {  
        if (start * 2 > LONG_MAX)  
            std::cout << "Stopping..." << std::endl;  
            break;  
  
        start *= 2;  
    }  
  
    std::cout << "Largest power of two representable by long: "  
              << start << std::endl;  
    // ...  
}
```

Debugging in `code expert`

```
// ...
// Finding the smallest negative power of 2 representable by long
start = -2;
while (true) {
    if (start * 2 < LONG_MIN)
        std::cout << "Stopping..." << std::endl;
        break;

    start *= 2;
}

std::cout << "Smallest negative power of two representable by long: "
          << start << std::endl;

return 0;
}
```

Debugging in **code expert**

Aufgabe

Öffnet das Beispiel "Debugging in CodeExpert" auf **code expert**

Description

Find the largest (N) and smallest negative (n) power of two that is representable by **long**. Formally that is:

$$N = \max(\{2^i : i \in \mathbb{N} \text{ and } 2^i \text{ is representable by } \mathbf{long}\})$$

$$n = \min(\{-2^i : i \in \mathbb{N} \text{ and } -2^i \text{ is representable by } \mathbf{long}\})$$

Debugging in **code expert**

Frage

Wie soll das Programm funktionieren?

Debugging in **code expert**

Frage

Wie soll das Programm funktionieren?

Antwort

2er-Potenzen berechnen, bis man den maximalen Wert von **long** erreicht

Debugging in **code expert**

Frage

Wie soll das Programm funktionieren?

Antwort

2er-Potenzen berechnen, bis man den maximalen Wert von **long** erreicht

Frage

Es gibt einen Syntaxfehler im Code. Kann jemand erkennen, was falsch ist?

Debugging in **code expert**

Frage

Wie soll das Programm funktionieren?

Antwort

2er-Potenzen berechnen, bis man den maximalen Wert von **long** erreicht

Frage

Es gibt einen Syntaxfehler im Code. Kann jemand erkennen, was falsch ist?

Antwort

die {} fehlen nach dem **if**-Zweig.

Debugging in **code expert**

Frage

Was ist an diesem Programm noch falsch?

Debugging in `code expert`

Frage

Was ist an diesem Programm noch falsch?

Antwort

Der Ausdruck `*2` führt zu einem **overflow**, was ein undefiniertes Verhalten für `signed int` ist. Hier führt es zu einer Endlosschleife!

Debugging in `code expert`

Frage

Was ist an diesem Programm noch falsch?

Antwort

Der Ausdruck `*2` führt zu einem **overflow**, was ein undefiniertes Verhalten für `signed int` ist. Hier führt es zu einer Endlosschleife!

Frage

Wie können wir also den Code korrigieren?

Debugging in `code expert` - Lösung

```
int main() {
    long start = 2;
    // Finding the largest power of 2 representable by long
    while (true) {
        if (start > LONG_MAX / 2) {
            std::cout << "Stopping..." << std::endl;
            break;
        }
        start *= 2;
    }

    std::cout << "Largest power of two representable by long: "
              << start << std::endl;
    // ...
}
```

Debugging in `code expert` - Lösung

```
// ...
// Finding the smallest negative power of 2 representable by long
start = -2;
while (true) {
    if (start < LONG_MIN / 2) {
        std::cout << "Stopping..." << std::endl;
        break;
    }
    start *= 2;
}

std::cout << "Smallest negative power of two representable by long: "
          << start << std::endl;

return 0;
}
```

Fragen/Unklarheiten?

7. Selection Statements

Even Numbers

Even Numbers

Aufgabe

Öffnet das Beispiel "Even Numbers" auf [code expert](#) und versucht es zu lösen

Description

Write a program that accepts an input value n (as an `int`). If the input value is non-negative, it outputs the biggest even number m that does not exceed the input value. If the input value is negative, then the program should output 0.

Input A whole number n

Output

- If $n \geq 0$, output the biggest even number $m \leq n$
- If $n < 0$, output 0

Examples

$2 \rightarrow 2$, $13 \rightarrow 12$, $43 \rightarrow 42$

Even Numbers - Lösung

Even Numbers - Lösung

```
#include <iostream>
int main () {

    int n;
    std::cin >> n;
    if ( n < 0 ) {
        n = 0;
    }
    if (n % 2 == 1) {
        n = n - 1;
    }

    std::cout << n << std::endl;

    return 0;
}
```

Fragen/Unklarheiten?

Alternative Basen

Zahlen in anderen Basen funktionieren konzeptionell genau gleich

Alternative Basen

$$F_{16} \neq 16_{10}$$

Zahlen in anderen Basen funktionieren konzeptionell genau gleich

binär	hexadezimal	dezimal
1111	F	15
1(1111) C_{16} 12_{10} 5	1F	31
11 0111 1100 0101	37C5	14'277
1010'1100'1101'1100	ACDC	44'252
1'0000'0000'0000'0000	1'0000	65'536
1010'1111'1111'1110'0000'1000'0001'0101	ÄFFE'0815	2'952'661'013

$$16 \cdot 16^7 \dots 8 \cdot 16^2 + 9 \cdot 16^1 + 5 \cdot 16^0 =$$

Fragen/Unklarheiten?

Two's complement

Konversion von Dezimal zu Binär:

1. Nehme den Betrag der Zahl und schreibe die Binärrepräsentation davon
2. Bits flippen: 1 wird zu 0 und 0 wird zu 1
3. 1 addieren und den Overflow ignorieren

1111

0000
1

Two's complement

Konversion von Dezimal zu Binär:

1. Nehme den Betrag der Zahl und schreibe die Binärrepräsentation davon
2. Bits flippen: 1 wird zu 0 und 0 wird zu 1
3. 1 addieren und den Overflow ignorieren

Beispiel: -6

1. $+6$ lässt sich als 0110 schreiben
2. Nach Bit Flipping erhalten wir 1001
3. $1001 + 1 = 1010$

Two's complement

Tipps zu Two's Complement:

- Das erste Bit einer Zahl n heisst **most significant bit** und gibt das Vorzeichen an

$$\text{msb} \begin{cases} = 1 \rightarrow n < 0 \\ = 0 \rightarrow n \geq 0 \end{cases}$$

- Die Grösse (= Anzahl Bits) einer Zahl bestimmt, welche Zahlen dargestellt werden können
 - Mit n Bits können Zahlen von $-(2^{n-1})$ bis $2^{n-1} - 1$ dargestellt werden

Fragen/Unklarheiten?

9. Alte Prüfungsaufgabe

1 Typen und Werte / Types and values (9 Punkte)

Geben Sie für jeden der sechs Ausdrücke unten jeweils C++-Typ (0.5 P) und Wert (1 P) an! Nehmen Sie für Fließkommazahlen den Standard IEEE 754 an! *For each of the six expressions below, provide the C++ type (0.5 P) and value (1 P)! For floating point numbers, assume the IEEE 754 standard!*

Prüfung 2018 "Typen und Werte"

(a) $8 + 5 / 3$

1.5 P

Typ/*Type*

Wert/*Value*

(b) $1.0 * 0.1 == 0.1 * 1.0$

1.5 P

Typ/*Type*

Wert/*Value*

(c) $1e1 * 2e2$

1.5 P

Typ/*Type*

Wert/*Value*

Prüfung 2018 "Typen und Werte" – Lösung

(a) $8 + 5 / 3$

1.5 P

Typ/Type

int

Wert/Value

9

(b) $1.0 * 0.1 == 0.1 * 1.0$

1.5 P

Typ/Type

bool

Wert/Value

true

(c) $1e1 * 2e2$

1.5 P

Typ/Type

double

Wert/Value

2000

Prüfung 2018 "Typen und Werte"

(d) `5 / 2.0f + 5 / 2.0`

1.5 P

Typ/*Type*

Wert/*Value*

(e) `!false || true && false`

1.5 P

Typ/*Type*

Wert/*Value*

(f) `17 % 5u`

1.5 P

Typ/*Type*

Wert/*Value*

Prüfung 2018 "Typen und Werte" – Lösung

(d) `5 / 2.0f + 5 / 2.0`

1.5 P

Typ/Type

`double`

Wert/Value

`5`

(e) `!false || true && false`

1.5 P

Typ/Type

`bool`

Wert/Value

`true`

(f) `17 % 5u`

1.5 P

Typ/Type

`unsigned int`

Wert/Value

`2`

10. Tipps zu **code** expert

Task "two-complement integer representation (Optional)"

- nichts \rightarrow Basis 10
- **0b** \rightarrow Basis 2
- **0** \rightarrow Basis 8 (also **10** \neq **010**)
- **0x** \rightarrow Basis 16

Task "From decimal to binary representation"

- Denkt an letzte Übungsstunde...

Task "Fibonacci overflow check"

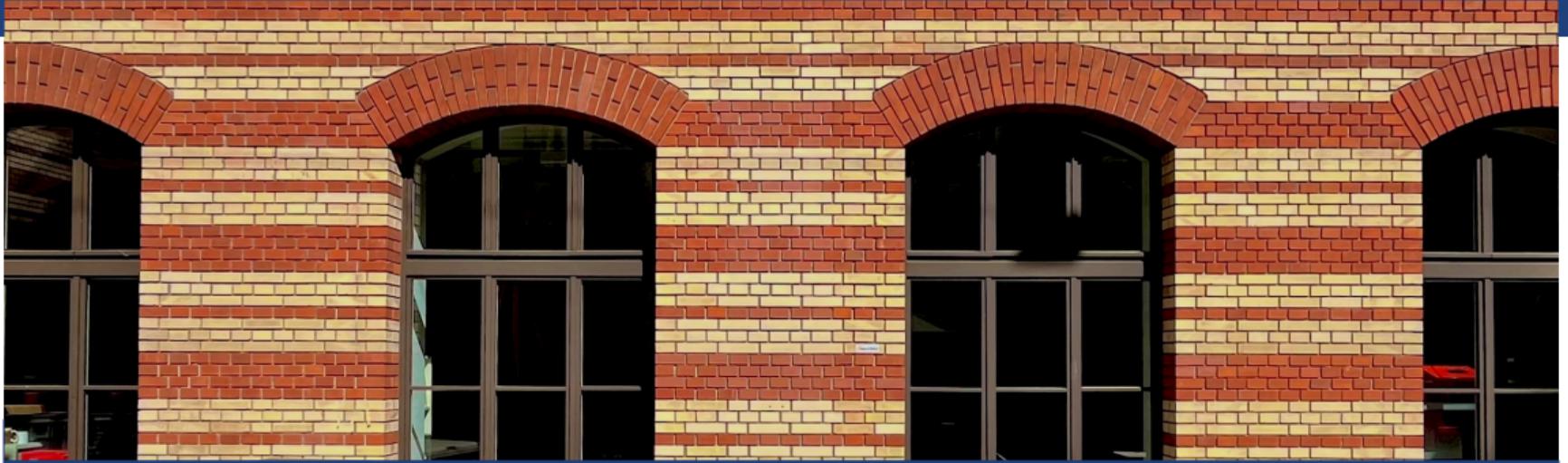
- **Most importantly:** *exit the print loop as soon as you detect that an overflow would occur., also darf die Addition nicht passieren!*

11. Outro

Allgemeine Fragen?

Bis zum nächsten Mal

Schöne Woche noch!



Übungsstunde — Informatik — 03

Adel Gavranović

Expressions, Loops, Reihen Berechnen, Scopes

Übersicht

Expressions

Loops

Reihen Berechnen

Scopes

Alte Prüfungsaufgabe



`n.ethz.ch/~agavranovic`

 Material

 Webpage

 Mail

Bevor es richtig losgeht...

Bevor es richtig losgeht...

- Wieso kommt ihr in *diese* Übungsstunde?

Bevor es richtig losgeht...

- Wieso kommt ihr in *diese* Übungsstunde?
- What about the english sessions?

2. Follow-up

Follow-up aus letzter Übungsstunde

Follow-up aus letzter Übungsstunde

- Habe ich etwas versäumt?

3. Feedback zu **code** expert

Allgemeines bezüglich **code expert**

Allgemeines bezüglich **code expert**

- Ich war *sehr* streng was die Korrekturen angeht. Das wird nicht immer so sein :)

Allgemeines bezüglich **code expert**

- Ich war *sehr* streng was die Korrekturen angeht. Das wird nicht immer so sein :)
- Nehmt euch (ein wenig) Zeit euren Code schön zu formatieren

Allgemeines bezüglich **code expert**

- Ich war *sehr* streng was die Korrekturen angeht. Das wird nicht immer so sein :)
- Nehmt euch (ein wenig) Zeit euren Code schön zu formatieren
- Falls beim Lösen Fragen aufkommen
 - schreibt sie bitte ganz oben im Code hin, damit ich sie direkt sehen kann (falls Sie nicht dringend sind)
 - oder schreibt mir eine E-Mail (falls Sie dringend sind) nachdem ihr bereits eine Abgabe gemacht habt und stellt sicher, dass ich die relevante Abgabe erkenne (mit einem Kommentar markieren oder so)

Allgemeines bezüglich **code expert**

- Ich war *sehr* streng was die Korrekturen angeht. Das wird nicht immer so sein :)
- Nehmt euch (ein wenig) Zeit euren Code schön zu formatieren
- Falls beim Lösen Fragen aufkommen
 - schreibt sie bitte ganz oben im Code hin, damit ich sie direkt sehen kann (falls Sie nicht dringend sind)
 - oder schreibt mir eine E-Mail (falls Sie dringend sind) nachdem ihr bereits eine Abgabe gemacht habt und stellt sicher, dass ich die relevante Abgabe erkenne (mit einem Kommentar markieren oder so)
- Woher haben so viele von euch eigentlich das `"/n"`?

Allgemeines bezüglich `code expert`

`const int a = 4;`

- Ich war *sehr* streng was die Korrekturen angeht. Das wird nicht immer so sein :)
- Nehmt euch (ein wenig) Zeit euren Code schön zu formatieren
- Falls beim Lösen Fragen aufkommen
 - schreibt sie bitte ganz oben im Code hin, damit ich sie direkt sehen kann (falls Sie nicht dringend sind)
 - oder schreibt mir eine E-Mail (falls Sie dringend sind) nachdem ihr bereits eine Abgabe gemacht habt und stellt sicher, dass ich die relevante Abgabe erkenne (mit einem Kommentar markieren oder so)
- Woher haben so viele von euch eigentlich das `"/n"`?
- Was bedeutet `const`?

`a = 7;` *Compiler error!*

Allgemeines bezüglich **code expert**

Allgemeines bezüglich **code expert**

- Meine Kommentare werden einheitlich auf Englisch geschrieben sein

Allgemeines bezüglich **code expert**

- Meine Kommentare werden einheitlich auf Englisch geschrieben sein
- 3/3 TA-points werden nicht häufig vergeben werden

Allgemeines bezüglich **code expert**

- Meine Kommentare werden einheitlich auf Englisch geschrieben sein
- 3/3 TA-points werden nicht häufig vergeben werden
- "Excellent" ist so ziemlich die höchste Auszeichnung

Allgemeines bezüglich **code expert**

- Meine Kommentare werden einheitlich auf Englisch geschrieben sein
- 3/3 TA-points werden nicht häufig vergeben werden
- "Excellent" ist so ziemlich die höchste Auszeichnung
- Im Editor auf **code expert** gibt es eine feine, graue Linie ganz rechts
 - Schaut, dass ihr die nie mit Code überschreitet
 - Auch nicht mit Kommentaren: spaltet sie lieber auf

```
// Das hier ist ein mehrzeiliger  
// Kommentar in C++, den man  
// noch immer gut lesen kann!
```

Fragen bezüglich **code expert** eurerseits?

4. Lernziele

Ziele für heute

- Komplexe Expressions, die arithmetische und Bool'sche Operatoren beinhalten, von Hand evaluieren können
- Mathematische Serien (Summen und Produkte) in C++ kodieren/implementieren können
- for**-, **while** und **do-while**-Schleifen tracen können
- Jede Art von Schleife in jede andere Art von Schleife umformen können

5. Zusammenfassung

6. Expressions

Types

Bisher behandelte Types

Bisher behandelte Types

- logic variables: `bool {false, true}`

Bisher behandelte Types

- logic variables: `bool {false, true}`
- integers: `unsigned int, int {-7, 2, 0}`

Bisher behandelte Types

- logic variables: `bool {false, true}`
- integers: `unsigned int, int {-7, 2, 0}`
- floating point numbers: `float, double {1.4, -4.3, 7.0}`

Bisher behandelte Types

- logic variables: `bool {false, true}`
- integers: `unsigned int, int {-7, 2, 0}`
- floating point numbers: `float, double {1.4, -4.3, 7.0}`

Manchmal sind mehrere Types in einer Expression.
Wie interagieren verschiedene Typen miteinander?

Bisher behandelte Types

- logic variables: `bool {false, true}`
- integers: `unsigned int, int {-7, 2, 0}`
- floating point numbers: `float, double {1.4, -4.3, 7.0}`

Manchmal sind mehrere Types in einer Expression.
Wie interagieren verschiedene Typen miteinander?

Generalitätsreihenfolge der Typen

Bisher behandelte Types

- logic variables: `bool {false, true}`
- integers: `unsigned int, int {-7, 2, 0}`
- floating point numbers: `float, double {1.4, -4.3, 7.0}`

Manchmal sind mehrere Types in einer Expression.
Wie interagieren verschiedene Typen miteinander?

Generalitätsreihenfolge der Typen

`bool <`

Bisher behandelte Types

- logic variables: `bool {false, true}`
- integers: `unsigned int, int {-7, 2, 0}`
- floating point numbers: `float, double {1.4, -4.3, 7.0}`

Manchmal sind mehrere Types in einer Expression.
Wie interagieren verschiedene Typen miteinander?

Generalitätsreihenfolge der Typen

```
bool < int < unsigned int <
```

Bisher behandelte Types

- logic variables: `bool {false, true}`
- integers: `unsigned int, int {-7, 2, 0}`
- floating point numbers: `float, double {1.4, -4.3, 7.0}`

Manchmal sind mehrere Types in einer Expression.
Wie interagieren verschiedene Typen miteinander?

Generalitätsreihenfolge der Typen

`bool < int < unsigned int < float < double`

Types konvertieren immer zum generellsten Type der Expression

Wie man sich Types vorstellen kann

Type (literal)

Approximiert

Wie man sich Types vorstellen kann

Type (literal)

`bool`

Approximiert

`{false, true}`

Wie man sich Types vorstellen kann

Type (literal)	Approximiert
<code>bool</code>	<code>{false, true}</code>
<code>unsigned int (u)</code>	\mathbb{N}

Wie man sich Types vorstellen kann

Type (literal)	Approximiert
<code>bool</code>	<code>{false, true}</code>
<code>unsigned int (u)</code>	\mathbb{N}
<code>int</code>	\mathbb{Z}

Wie man sich Types vorstellen kann

Type (literal)	Approximiert
<code>bool</code>	<code>{false, true}</code>
<code>unsigned int (u)</code>	\mathbb{N}
<code>int</code>	\mathbb{Z}
<code>float (f)</code>	\mathbb{R}

Wie man sich Types vorstellen kann

Type (literal)

`bool`

`unsigned int (u)`

`int`

`float (f)`

`double`

Approximiert

`{false, true}`

\mathbb{N}

\mathbb{Z}

\mathbb{R}

\mathbb{R} , aber *double* Präzision

Types evaluieren I

```
std::cout << 5.0/2 << std::endl;  
// what type and value will this return and why?
```

Types evaluieren I

```
std::cout << 5.0/2 << std::endl;  
// what type and value will this return and why?
```

Lösung

double, 2.5, weil die **int** 2 zuerst in eine **double** 2.0 konvertiert wird, um diese Expression zu berechnen.

Types evaluieren II

```
std::cout << (1/2)*5.0/2 << std::endl;  
// what type and value will this return and why?
```

Types evaluieren II

```
std::cout << (1/2)*5.0/2 << std::endl;  
// what type and value will this return and why?
```

Lösung

double, 0, weil zuerst die linke Expression $1/2$ evaluiert wird, welche zu 0 evaluiert (Integer-Division). Der Rest ist trivial, weil $0*$ anything evaluiert zu 0. Aber diese 0 wird vom Type **double** sein.

Literale

Literale

Es gibt bestimmte Buchstaben, die der Compiler mit bestimmten Types verbindet. Wenn ihr dem Compiler sagen möchtest *"Hey, don't treat this 2.0 as a **double**, but instead as a **float**"* müsst ihr ein `f` am Ende des Werts hinzufügen. Etwa so:

Literale

Es gibt bestimmte Buchstaben, die der Compiler mit bestimmten Types verbindet. Wenn ihr dem Compiler sagen möchtest "Hey, don't treat this *2.0* as a *double*, but instead as a *float*" müsst ihr ein `f` am Ende des Werts hinzufügen. Etwa so:

```
| std::cout << (5/2)*5.0f/2 << std::endl;
```

Types evaluieren III

```
std::cout << (5/2)*5.0f/2 << std::endl;  
// what type and value will this return and why?
```

Types evaluieren III

```
std::cout << (5/2)*5.0f/2 << std::endl;  
// what type and value will this return and why?
```

Lösung

float, 5.0, (kann als 5.0f geschrieben werden).

Zuerst, wird 5/2 evaluiert, was zu 2 wird (integer division). Dann wird 2.0f*5.0f berechnet: Die int 2 wurde zu einer float 2, weil float der generellere Type (in dieser Expression) ist. Dito für /2 später.

Exercise I

1. Welche der folgenden Zeichenfolgen sind keine C++ Expression, und warum nicht? Hierbei seien `x` und `y` Variablen vom Typ `int`.
 - a) `(y++ < 0 && y < 0) + 2.0`
 - b) `y = (x++ = 3)`
 - c) `3.0 + 3 - 4 + 5`
 - d) `5 % 4 * 3.0 + true * x++`
2. Für alle gültigen Expression, die oben identifiziert wurden, entscheide, ob es sich um l-Werte oder r-Werte handelt und begründe deine Entscheidung.
3. Determine the values of the expressions and explain how these values are obtained. Assume that initially `x == 1` and `y == -1`.

Expression Evaluation - Lösungen a)

`(y++ < 0 && y < 0) + 2.0`

Expression Evaluation - Lösungen a)

```
(y++ < 0 && y < 0) + 2.0
```

```
(-1 < 0 && y < 0) + 2.0 // after this step: y==0
```

Expression Evaluation - Lösungen a)

```
(y++ < 0 && y < 0) + 2.0
```

```
(-1 < 0 && y < 0) + 2.0 // after this step: y==0
```

```
(true && y < 0) + 2.0
```

Expression Evaluation - Lösungen a)

```
(y++ < 0 && y < 0) + 2.0
```

```
(-1 < 0 && y < 0) + 2.0 // after this step: y==0
```

```
(true && y < 0) + 2.0
```

```
(true && false) + 2.0
```

Expression Evaluation - Lösungen a)

```
(y++ < 0 && y < 0) + 2.0
```

```
(-1 < 0 && y < 0) + 2.0 // after this step: y==0
```

```
(true && y < 0) + 2.0
```

```
(true && false) + 2.0
```

```
(false) + 2.0
```

Expression Evaluation - Lösungen a)

```
(y++ < 0 && y < 0) + 2.0
```

```
(-1 < 0 && y < 0) + 2.0 // after this step: y==0
```

```
(true && y < 0) + 2.0
```

```
(true && false) + 2.0
```

```
(false) + 2.0
```

```
0.0 + 2.0
```

Expression Evaluation - Lösungen a)

```
(y++ < 0 && y < 0) + 2.0
```

```
(-1 < 0 && y < 0) + 2.0 // after this step: y==0
```

```
(true && y < 0) + 2.0
```

```
(true && false) + 2.0
```

```
(false) + 2.0
```

```
0.0 + 2.0
```

```
2.0
```

Expression Evaluation - Lösungen a)

```
(y++ < 0 && y < 0) + 2.0
```

```
(-1 < 0 && y < 0) + 2.0 // after this step: y==0
```

```
(true && y < 0) + 2.0
```

```
(true && false) + 2.0
```

```
(false) + 2.0
```

```
0.0 + 2.0
```

```
2.0
```

r-Wert

Expression Evaluation - Lösungen b)

`y = (x++ = 3)`

Expression Evaluation - Lösungen b)

`y = (x++ = 3)`

Invalid

Expression Evaluation - Lösungen c)

$$3.0 + 3 - 4 + 5$$

Expression Evaluation - Lösungen c)

$$3.0 + 3 - 4 + 5$$

$$((3.0 + 3) - 4) + 5$$

Expression Evaluation - Lösungen c)

$$3.0 + 3 - 4 + 5$$

$$((3.0 + 3) - 4) + 5$$

$$((3.0 + 3.0) - 4) + 5$$

Expression Evaluation - Lösungen c)

$$3.0 + 3 - 4 + 5$$

$$((3.0 + 3) - 4) + 5$$

$$((3.0 + 3.0) - 4) + 5$$

$$(6.0 - 4) + 5$$

Expression Evaluation - Lösungen c)

$$3.0 + 3 - 4 + 5$$

$$((3.0 + 3) - 4) + 5$$

$$((3.0 + 3.0) - 4) + 5$$

$$(6.0 - 4) + 5$$

$$(6.0 - 4.0) + 5$$

Expression Evaluation - Lösungen c)

$$3.0 + 3 - 4 + 5$$

$$((3.0 + 3) - 4) + 5$$

$$((3.0 + 3.0) - 4) + 5$$

$$(6.0 - 4) + 5$$

$$(6.0 - 4.0) + 5$$

$$2.0 + 5$$

Expression Evaluation - Lösungen c)

$$3.0 + 3 - 4 + 5$$

$$((3.0 + 3) - 4) + 5$$

$$((3.0 + 3.0) - 4) + 5$$

$$(6.0 - 4) + 5$$

$$(6.0 - 4.0) + 5$$

$$2.0 + 5$$

$$2.0 + 5.0$$

Expression Evaluation - Lösungen c)

$$3.0 + 3 - 4 + 5$$

$$((3.0 + 3) - 4) + 5$$

$$((3.0 + 3.0) - 4) + 5$$

$$(6.0 - 4) + 5$$

$$(6.0 - 4.0) + 5$$

$$2.0 + 5$$

$$2.0 + 5.0$$

$$7.0$$

Expression Evaluation - Lösungen c)

$$3.0 + 3 - 4 + 5$$

$$((3.0 + 3) - 4) + 5$$

$$((3.0 + 3.0) - 4) + 5$$

$$(6.0 - 4) + 5$$

$$(6.0 - 4.0) + 5$$

$$2.0 + 5$$

$$2.0 + 5.0$$

$$7.0$$

r-Wert

Expression Evaluation - Lösungen d)

5 % 4 * 3.0 + true * x++

1 * 3.0
3.0d * 1.0
4.0d * 1
4.0

Expression Evaluation - Lösungen d)

```
5 % 4 * 3.0 + true * x++
```

```
((5 % 4) * 3.0) + (true * (x++))
```

Expression Evaluation - Lösungen d)

```
5 % 4 * 3.0 + true * x++
```

```
((5 % 4) * 3.0) + (true * (x++))
```

```
(1 * 3.0) + (true * (x++))
```

Expression Evaluation - Lösungen d)

```
5 % 4 * 3.0 + true * x++
```

```
((5 % 4) * 3.0) + (true * (x++))
```

```
(1 * 3.0) + (true * (x++))
```

```
(1.0 * 3.0) + (true * (x++))
```

Expression Evaluation - Lösungen d)

```
5 % 4 * 3.0 + true * x++
```

```
((5 % 4) * 3.0) + (true * (x++))
```

```
(1 * 3.0) + (true * (x++))
```

```
(1.0 * 3.0) + (true * (x++))
```

```
3.0 + (true * (x++))
```

Expression Evaluation - Lösungen d)

```
5 % 4 * 3.0 + true * x++
```

```
((5 % 4) * 3.0) + (true * (x++))
```

```
(1 * 3.0) + (true * (x++))
```

```
(1.0 * 3.0) + (true * (x++))
```

```
3.0 + (true * (x++))
```

```
3.0 + (true * 1)
```

Expression Evaluation - Lösungen d)

```
5 % 4 * 3.0 + true * x++
```

```
((5 % 4) * 3.0) + (true * (x++))
```

```
(1 * 3.0) + (true * (x++))
```

```
(1.0 * 3.0) + (true * (x++))
```

```
3.0 + (true * (x++))
```

```
3.0 + (true * 1)
```

```
3.0 + (1 * 1)
```

Expression Evaluation - Lösungen d)

```
5 % 4 * 3.0 + true * x++
```

```
((5 % 4) * 3.0) + (true * (x++))
```

```
(1 * 3.0) + (true * (x++))
```

```
(1.0 * 3.0) + (true * (x++))
```

```
3.0 + (true * (x++))
```

```
3.0 + (true * 1)
```

```
3.0 + (1 * 1)
```

```
3.0 + 1
```

Expression Evaluation - Lösungen d)

```
5 % 4 * 3.0 + true * x++
```

```
((5 % 4) * 3.0) + (true * (x++))
```

```
(1 * 3.0) + (true * (x++))
```

```
(1.0 * 3.0) + (true * (x++))
```

```
3.0 + (true * (x++))
```

```
3.0 + (true * 1)
```

```
3.0 + (1 * 1)
```

```
3.0 + 1
```

```
3.0 + 1.0
```

Expression Evaluation - Lösungen d)

```
5 % 4 * 3.0 + true * x++
```

```
((5 % 4) * 3.0) + (true * (x++))
```

```
(1 * 3.0) + (true * (x++))
```

```
(1.0 * 3.0) + (true * (x++))
```

```
3.0 + (true * (x++))
```

```
3.0 + (true * 1)
```

```
3.0 + (1 * 1)
```

```
3.0 + 1
```

```
3.0 + 1.0
```

```
4.0
```

Expression Evaluation - Lösungen d)

```
5 % 4 * 3.0 + true * x++
```

```
((5 % 4) * 3.0) + (true * (x++))
```

```
(1 * 3.0) + (true * (x++))
```

```
(1.0 * 3.0) + (true * (x++))
```

```
3.0 + (true * (x++))
```

```
3.0 + (true * 1)
```

```
3.0 + (1 * 1)
```

```
3.0 + 1
```

```
3.0 + 1.0
```

```
4.0
```

r-Wert

Loop Correctness

Kann man beim Laufen dieses Programms den Unterschied zwischen den Ausgaben dieser drei Schleifen feststellen? Wenn ja, wie? Nehmen wir an, dass `n` eine Variable vom Typ `unsigned int` ist, deren Wert eingegeben wird.

Loop Correctness

Kann man beim Laufen dieses Programms den Unterschied zwischen den Ausgaben dieser drei Schleifen feststellen? Wenn ja, wie? Nehmen wir an, dass `n` eine Variable vom Typ `unsigned int` ist, deren Wert eingegeben wird.

```
////////////////////////////////////  
unsigned int n;  
std::cin >> n;  
unsigned int i;  
  
// loop 1 //////////////////////////////////////  
for (i = 1; i <= n; ++i) {  
    std::cout << i << "\n";  
}
```

```
// loop 2 //////////////////////////////////////  
i = 0;  
while (i < n) {  
    std::cout << ++i << "\n";  
}  
  
// loop 3 //////////////////////////////////////  
i = 1;  
do {  
    std::cout << i++ << "\n";  
} while (i <= n);
```

Schleifenkorrektheit - Lösung

Lösung

Es gibt die folgenden Unterschiede:

- Im Gegensatz zu den Schleifen 1 und 2 wird in Schleife 3 für die Eingabe $n == 0$ 1 ausgegeben, da die Anweisung in einer **do**-Schleife immer einmal ausgeführt wird, bevor die Bedingung geprüft wird
- Wenn n die grösstmögliche ganze Zahl ist, dann können die Schleifen 1 und 3 unendlich sein, weil die Bedingung $i \leq n$ für alle möglichen i wahr sein wird

Fragen/Unklarheiten?

7. Loops

for → while

```
// TASK: Convert the following  
// for-loop into an  
// equivalent while-loop:
```

```
for (int i = 0; i < n; ++i) {  
    // BODY  
}
```

condition

```
int i = 0;  
while ( i < n ) {  
    // Body (i)  
    ++i;  
}
```

for → while

```
// TASK: Convert the following
// for-loop into an
// equivalent while-loop:

for (int i = 0; i < n; ++i) {
    // BODY
}
```

```
// SOLUTION

int i = 0;

while(i < n){
    // BODY
    ++i;
}
```

while → for

```
// TASK: Convert the following
// while-loop into an
// equivalent for-loop:

while(condition){
    // BODY
}
```

while → for

```
// TASK: Convert the following
// while-loop into an
// equivalent for-loop:

while(condition){
    // BODY
}
```

```
// SOLUTION

for(;condition;){
    // BODY
}
```

do-while → for

```
// TASK: Convert the following
// do-while-loop into an
// equivalent for-loop:

do{
    // BODY
}while(condition)
```

do-while → for

```
// TASK: Convert the following
// do-while-loop into an
// equivalent for-loop:

do{
    // BODY
}while(condition)
```

```
// SOLUTION

// BODY
for(;condition;){
    // BODY
}
```

Fragen/Unklarheiten?

8. Reihen Berechnen

Von Summe zur Schleife

Mathematische Summen können zu Loops umgewandelt werden

$$\sum_{i=0}^n f(i)$$

Von Summe zur Schleife

Mathematische Summen können zu Loops umgewandelt werden

$$\sum_{i=0}^n \underline{f(i)}$$

f(i) ...
{ ... }

Wird zu

```
int n = 0;  
int sum = 0;  
std::cin >> n;  
for(int i = 0; i <= n; i++){  
    sum += f(i);  
}
```

Aufwärmübungen

Betrachten wir die Formel

$$\frac{1}{n!}$$

Aufwärmübungen

Betrachten wir die Formel

$$\frac{1}{n!} = \frac{1}{1} \cdot \frac{1}{2} \cdots \frac{1}{n}$$

Aufwärmübungen

Betrachten wir die Formel

$$\frac{1}{n!} = \frac{1}{1} \cdot \frac{1}{2} \cdots \frac{1}{n}$$

Wie könnte man dies als
("multiplikative") Reihe umsetzen?

$$\frac{1}{n!}$$

Aufwärmübungen

Betrachten wir die Formel

$$\frac{1}{n!} = \frac{1}{1} \cdot \frac{1}{2} \cdots \frac{1}{n}$$

Wie könnte man dies als
("multiplikative") Reihe umsetzen?

$$\frac{1}{n!} = \prod_{i=1}^n \frac{1}{i}$$

Wie verwandeln wir dieses Stück
Mathematik in ein Stück C++-Code?

Aufwärmübungen

Betrachten wir die Formel

$$\frac{1}{n!} = \frac{1}{1} \cdot \frac{1}{2} \cdot \dots \cdot \frac{1}{n}$$

Wie könnte man dies als
("multiplikative") Reihe umsetzen?

$$\frac{1}{n!} = \prod_{i=1}^n \frac{1}{i}$$

Wie verwandeln wir dieses Stück
Mathematik in ein Stück C++-Code?

```
int main(){  
  
    int n;           // user input  
    double result;  // main output  
  
    [your code goes  
     here!]  
  
    std::cout << result  
               << std::endl;  
  
    return 0;  
}
```

Aufwärmübungen - Lösungsbeispiel

```
int main(){
    int n;
    double result = 1;
    int i = 1;

    std::cin >> n;

    while(i <= n){
        result = result/i;
        i++;
    }

    std::cout << result << std::endl;

    return 0;
}
```

Von Reihe zur Schleife

Taylor Series auf **code expert**

Schreibe ein Programm, dass $\sin(x)$ bis auf sechs Stellen berechnet.

Tipp: Welchen Loop sollte man hierfür verwenden?

Tipp: MacLaurin-Reihe verwenden.

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

Von Reihe zur Schleife

Taylor Series auf **code expert**

Schreibe ein Programm, dass $\sin(x)$ bis auf sechs Stellen berechnet.

Tipp: Welchen Loop sollte man hierfür verwenden?

Tipp: MacLaurin-Reihe verwenden.

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

Aufgabe

- Mit Stift und Papier versuchen

Von Reihe zur Schleife

Taylor Series auf **code expert**

Schreibe ein Programm, dass $\sin(x)$ bis auf **sechs Stellen berechnet**.

Tipp: Welchen Loop sollte man hierfür verwenden?

Tipp: MacLaurin-Reihe verwenden.

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1}$$

Aufgabe

- Mit Stift und Papier versuchen
- Mit Person neben euch versuchen in **code expert** zu implementieren

Von Reihe zur Schleife - Lösung

Von Reihe zur Schleife - Lösung

```
#include <iostream>

int main () {

    double x;
    std::cin >> x;

    double numtor = x;
    double denomtor = 1;

    double sum = x;
    double term;
    double term_abs;
    int n = 1;
```

```
do {
    numtor *= -(x * x);
    denomtor *= (2 * n) * (2 * n + 1);
    term = numtor / denomtor;
    sum += term;
    if (term < 0) {
        term_abs = -term;
    } else {
        term_abs = term;
    }
    ++n;
} while (term_abs > 0.000001);

std::cout << sum << std::endl;
return 0;
}
```

Fragen/Unklarheiten?

9. Scopes

Scopes

Frage

In der Vorlesung von dieser Woche wurde ein neues Konzept eingeführt: "Variable Scopes". Was sind "Variable Scopes" und warum brauchen wir sie?

Scopes

Frage

In der Vorlesung von dieser Woche wurde ein neues Konzept eingeführt: "Variable Scopes". Was sind "Variable Scopes" und warum brauchen wir sie?

Antwort

Scopes definieren die Codesegmente unseres Programms, in denen eine Variable (l-Wert) existiert. Der Scope einer Variablen beginnt an der Stelle, an der sie definiert wurde, und endet am Ende des Blocks, in dem sie definiert wurde. Zum Beispiel:

```
if (x < 7){  
    int a = 8;           // <-- a's variable scope BEGINS here!  
    std::cout << a;     // Fine, prints 8.  
}                       // <-- a's variable scope ENDS here!  
std::cout << a;        // Compiler error, a does not exist.
```

Bug?

Ein vermeindlicher Weg, den Fehler zu beheben, wäre:

```
int a = 2;

if (x < 7) {
    int a = 8;
    std::cout << a;
}

std::cout << a;
```

Frage

Was wird dieses Programm wiedergeben wenn $x==2$?

Bug?

Ein vermeindlicher Weg, den Fehler zu beheben, wäre:

```
int a = 2;

if (x < 7) {
    int a = 8;
    std::cout << a;
}

std::cout << a;
```

Frage

Was wird dieses Programm wiedergeben wenn $x==2$?

Antwort

Es wird 82 wiedergegeben.

Bug?

Ein vermeindlicher Weg, den Fehler zu beheben, wäre:

```
int a = 2;

if (x < 7) {
    int a = 8;
    std::cout << a;
}

std::cout << a;
```

Frage

Was wird dieses Programm wiedergeben wenn $x==2$?

Antwort

Es wird 82 wiedergegeben.

Warum? Siehe [Program Tracing Guide](#)

Bug?

Frage

Was ist der Scope von `sum`, `i` und `a` im folgenden Beispiel?

```
int sum = 0;

for (int i = 0; i < 5; ++i) {
    int a;
    std::cin >> a;
    sum += a;
}
```

Antwort

Bug?

Frage

Was ist der Scope von `sum`, `i` und `a` im folgenden Beispiel?

```
int sum = 0;

for (int i = 0; i < 5; ++i) {
    int a;
    std::cin >> a;
    sum += a;
}
```

Antwort

`sum` (Mindestens) das gesamte Snippet

Bug?

Frage

Was ist der Scope von `sum`, `i` und `a` im folgenden Beispiel?

```
int sum = 0;

for (int i = 0; i < 5; ++i) {
    int a;
    std::cin >> a;
    sum += a;
}
```

Antwort

`sum` (Mindestens) das gesamte Snippet

`i` Die gesamte `for`-Schleife

Bug?

Frage

Was ist der Scope von `sum`, `i` und `a` im folgenden Beispiel?

```
int sum = 0;

for (int i = 0; i < 5; ++i) {
    int a;
    std::cin >> a;
    sum += a;
}
```

Antwort

`sum` (Mindestens) das gesamte Snippet

`i` Die gesamte `for`-Schleife

`a` Eine Schleifeniteration. Anders gesagt: Am Anfang des Schleifenkörpers hat `a` *nicht* garantiert den Wert, den es am Ende des Schleifenkörpers in der vorherigen Schleifeniteration hatte.

Fragen/Unklarheiten?

10. Alte Prüfungsaufgabe

By the way...

By the way...

Falls ihr selbst durch alte Prüfungen gehen wollt, sind sie  unter "Informatik I, Computer Science Introduction, C++" auffindbar.

Eure Prüfung wird denen mit "CSE" wahrscheinlich am ähnlichsten sein.

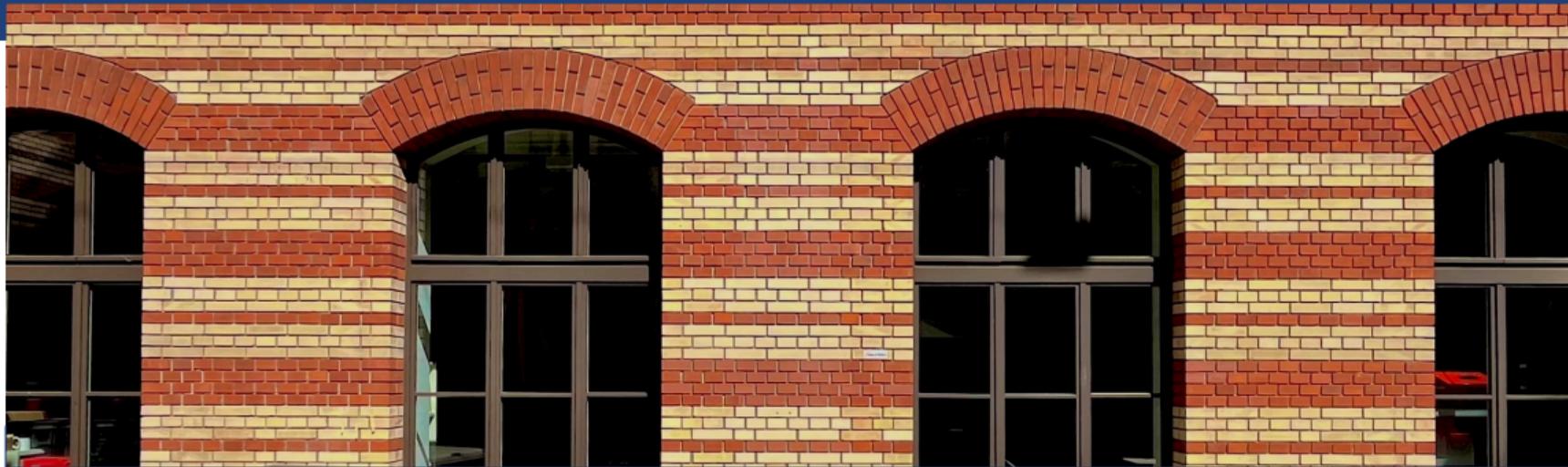
Das Passwort sollte überall **Informatik** sein.

11. Outro

Allgemeine Fragen?

Bis zum nächsten Mal

Schöne Woche noch!



Übungsstunde — Informatik — 05

Adel Gavranović

code expert -Änderungen, assert, PRE und POST, Funktionen, Headerfiles, Namespaces

Übersicht

code expert Neuerungen

assert

PRE und POST

Functions

Exam Question

Headers und Namespaces

Stepwise Refinement

Alte Prüfungsfragen



`n.ethz.ch/~agavranovic`

 Material

 Webpage

 Mail

1. Intro

- Hoffe euch hat die Session mit Ioana¹ gefallen! :)

¹  Ioana's Webpage

2. Follow-up

Follow-up aus letzter Übungsstunde

Follow-up aus letzter Übungsstunde

- Gab es noch offene Fragen bezüglich letzter Übungsstunde?

Follow-up aus letzter Übungsstunde

- Gab es noch offene Fragen bezüglich letzter Übungsstunde?
- Who liked it better in english?

3. Feedback zu **code** expert

Allgemeines bezüglich **code expert**

Allgemeines bezüglich **code expert**

Allgemeines bezüglich **code expert**

- Ihr müsst keine `<>` setzen beim beantworten der Fragen. Schaut euch das kompilierte markdown-file einfach beim schreiben an :)

Allgemeines bezüglich **code expert**

- Ihr müsst keine `<>` setzen beim beantworten der Fragen. Schaut euch das kompilierte markdown-file einfach beim schreiben an :)
- Bitte löscht die Aufgabenbeschreibung, Tips und jegliche TODOs aus euren Abgaben (sonst gibt's Punkteabzug) ausser ihr macht was sinnvolles mit ihnen

Allgemeines bezüglich **code expert**

- Ihr müsst keine `<>` setzen beim beantworten der Fragen. Schaut euch das kompilierte markdown-file einfach beim schreiben an :)
- Bitte löscht die Aufgabenbeschreibung, Tips und jegliche TODOs aus euren Abgaben (sonst gibt's Punkteabzug) ausser ihr macht was sinnvolles mit ihnen
- Falls jemand denkt, einen guten Grund (z.B. Militärdienst) zu haben, wieso eine verspätete Abgabe noch gewertet werden soll, so soll die Person mir mit kontaktieren (am besten via Mail)

Informationen bezüglich Bonusaufgabe I

Informationen bezüglich Bonusaufgabe I

- 10 Reguläre Tests ($\approx 71\%$)
- 4 Versteckte Tests

Informationen bezüglich Bonusaufgabe I

- 10 Reguläre Tests ($\approx 71\%$)
- 4 Versteckte Tests
- Keine TA-Punkte von mir (aber wenn ihr wollt, dass ich den Code verstehe, gebt euch beim Formatting Mühe)

Informationen bezüglich Bonusaufgabe I

- 10 Reguläre Tests ($\approx 71\%$)
- 4 Versteckte Tests
- Keine TA-Punkte von mir (aber wenn ihr wollt, dass ich den Code verstehe, gebt euch beim Formatting Mühe)
- Exercises ab "Perpetual Calendar" sammeln Punkte für die Bonusaufgabe II

Allgemeines bezüglich **code expert**

Allgemeines bezüglich **code expert**

- Sind alle mit der Art des Feedbacks zufrieden?

Allgemeines bezüglich **code expert**

- Sind alle mit der Art des Feedbacks zufrieden?
- Gibt es Worte/Ausdrücke im Feedback die ihr nicht versteht?

Allgemeines bezüglich **code expert**

- Sind alle mit der Art des Feedbacks zufrieden?
- Gibt es Worte/Ausdrücke im Feedback die ihr nicht versteht?
- Was könnte ich besser machen?

Fragen bezüglich **code expert** eurerseits?

4. Lernziele

Ziele für Heute

Lernziele

2 [Program Tracing Guide](#)

3 [Style Guide](#)

Ziele für Heute

Lernziele

- in der Lage sein, Vor- und Nachbedingungen (PRE and POST conditions) für Funktionen zu schreiben
- in der Lage sein, Aufgaben mittels "stepwise refinement" zu lösen
- in der Lage sein, `assert()` richtig anzuwenden
- in der Lage sein, Code, der Funktionen aufruft, zu `tracen`²
- Verstehen, was die Gefahren von `namespace` sind
- in der Lage sein, verständlichen und schönen Code zu schreiben³

²  [Program Tracing Guide](#)

³  [Style Guide](#)

5. Zusammenfassung

6. **code expert** Neuerungen

Neues Features: Versteckte Testfälle

Versteckte Testfälle

⁴Ohnehin verboten

Neues Features: Versteckte Testfälle

Versteckte Testfälle

- Ab Woche 7 enthalten einige [code]expert-Übungen versteckte Testfälle

⁴Ohnehin verboten

Neues Features: Versteckte Testfälle

Versteckte Testfälle

- Ab Woche 7 enthalten einige [code]expert-Übungen versteckte Testfälle
- Versteckte Testfälle zeigen bei Fehlern die erwartete Ausgabe nicht an, um Hard-Coding⁴ zu verhindern

⁴Ohnehin verboten

Neues Features: Versteckte Testfälle

Versteckte Testfälle

- Ab Woche 7 enthalten einige [code]expert-Übungen versteckte Testfälle
- Versteckte Testfälle zeigen bei Fehlern die erwartete Ausgabe nicht an, um Hard-Coding⁴ zu verhindern
- Versteckte Testfälle sind in Bonusübungen und Prüfungen üblich

⁴Ohnehin verboten

Neues Features: Versteckte Testfälle

Versteckte Testfälle

- Ab Woche 7 enthalten einige [code]expert-Übungen versteckte Testfälle
- Versteckte Testfälle zeigen bei Fehlern die erwartete Ausgabe nicht an, um Hard-Coding⁴ zu verhindern
- Versteckte Testfälle sind in Bonusübungen und Prüfungen üblich
- Übungen mit versteckten Testfällen und persistenter Eingabe werden in **code expert** mit "[hidden tests]" gekennzeichnet

⁴Ohnehin verboten

Neues Features: Persistente Eingabe

Persistente Eingabe

⁵Eselsbrücke: **f** wie file!

Neues Features: Persistente Eingabe

Persistente Eingabe

- Neues Feature eingeführt, um Eingaben (aus `input.txt`) über mehrere Durchläufe zu speichern und wiederzuverwenden

⁵Eselsbrücke: `f` wie file!

Neues Features: Persistente Eingabe

Persistente Eingabe

- Neues Feature eingeführt, um Eingaben (aus `input.txt`) über mehrere Durchläufe zu speichern und wiederzuverwenden
- Die persistente Eingabe hat keinen Einfluss auf Noten oder XP

⁵Eselsbrücke: `f` wie file!

Neues Features: Persistente Eingabe

Persistente Eingabe

- Neues Feature eingeführt, um Eingaben (aus `input.txt`) über mehrere Durchläufe zu speichern und wiederzuverwenden
- Die persistente Eingabe hat keinen Einfluss auf Noten oder XP
- Nutzung: `f` (ohne Leerzeichen) statt eigentliche Eingabe in Terminal eingeben⁵

⁵Eselsbrücke: `f` wie file!

Fragen/Unklarheiten?

7. assert

- Wir werden Exit-Codes sehen, mehr Infos hier⁶

Frage

- Wofür sind asserts nützlich?

Frage

- Wofür sind asserts nützlich?

Mögliche Antworten

- Um herauszufinden, wo genau ein Fehler passiert
- In langen Programmen, um den Überblick zu bewahren
- Falsche (User) Inputs sofort erkennen (hilft, *undefined behavior* zu vermeiden)
- Als eine Art der Code Dokumentation

assert Demo

code expert Code Example "Debugging with Assert"

Fragen/Unklarheiten?

8. PRE und POST

PRE und POST Conditions

```
// PRE: describes accepted input
// POST: describes expected output
int yourfunction(int a, int b){
    ...
}
```

PRE und POST Conditions

Frage: Was wären hier sinnvolle Conditions?

```
// PRE:  
// POST:  
double area(double height, double length){  
    return height*length;  
}
```

PRE und POST Conditions

Frage: Was wären hier sinnvolle Conditions?

```
// PRE:  
// POST:  
double area(double height, double length){  
    return height*length;  
}
```

Sie müssen nicht sehr detailliert sein, sie sollten beschreiben, was die Funktion erwartet und was sie ausgegeben wird (wenn der Input erwartungsgemäss war)

Fragen/Unklarheiten?

PRE und POST Conditions I

Bestimme sinnvolle PRE- und POST-conditions für die folgende Funktion

```
// PRE: ???  
// POST: ???  
double f(double i, double j, double k){  
    if(i > j){  
        if(i > k){return i;}  
        else {return k;}  
    } else {  
        if(j > k){return j;}  
        else {return k;}  
    }  
}
```

PRE und POST Conditions I (Lösung)

Mögliche Lösung

```
// PRE: (not needed)
// POST: return value is maximum of {i, j, k}
double f(double i, double j, double k){
    if(i > j){
        if(i > k){return i;}
        else {return k;}
    } else {
        if(j > k){return j;}
        else {return k;}
    }
}
```

PRE und POST Conditions II

Bestimme sinnvolle PRE- und POST-conditions für die folgende Funktion

```
// PRE: ???  
// POST: ???  
double g(int i, int j){  
    double r = 0.0;  
    for(int k = i; k <= j; k++){  
        r += 1.0 / k;  
    }  
    return r;  
}
```

PRE und POST Conditions II (Lösung)

Mögliche Lösung

```
// PRE: 0 not in [i, j] and i <= j < INT_MAX
// POST: return value is the sum 1/i + 1/(i+1) + ... + 1/j
double g(int i, int j){
    double r = 0.0;
    for(int k = i; k <= j; k++){
        r += 1.0 / k;
    }
    return r;
}
```

9. Functions

Output?

```
int f(int i){
    return i * i;
}

int g(int i){
    return i * f(i) * f(f(i));
}

int h(int i){
    std::cout << g(i) << "\n";
}
// ...
```

```
// ...
int main(){
    int i;
    std::cin >> i;
    h(i);
    return 0;
}
```

Was wird (mögliche Over- und Underflows vernachlässigt) der Output sein?

Output?

```
int f(int i){
    return i * i;
}

int g(int i){
    return i * f(i) * f(f(i));
}

void h(int i){
    std::cout << g(i) << "\n";
}
// ...
```

```
// ...
int main(){
    int i;
    std::cin >> i;
    h(i);
    return 0;
}
```

Was wird (mögliche Over- und Underflows vernachlässigt) der Output sein? **Lösung:** i^7

Fehlersuche

double g(double x);

```
double f(double x){  
    return g(2.0 * x);  
}
```

↓ bool

```
double g(double x){  
    return x % 2.0 == 0;  
}
```

void

```
double h(double x){  
    std::cout << result;  
} x?
```

// ...

```
// ...  
int main(){  
    double result = f(3.0);  
    h();  
    - fehl. arg!  
    return 0;  
}
```

Finde mindestens 3 Fehler in diesem Programm.

Fehlersuche (Lösung)

1. `g()` ist `f()` nicht bekannt, da der Scope von `g()` erst später anfängt
2. Es gibt keinen `%`-Operator für `double`
3. `h()` "sieht" die Variabel `result` nicht, da sie nicht im gleichen Scope ist
4. Die Funktion `h()` hat keine Rückgabe, obwohl es eine braucht
5. `h()` wird ohne Argument aufgerufen

Anzahl Divisoren

Schreibe eine Funktion `number_of_divisors()`, die `int n` als Argument entgegennimmt und die Anzahl Divisoren von n zurückgibt (inklusive 1 und n)

```
// PRE: 0 < n < MAX_INT
// POST: returns number of divisors of n (incl. 1 and n)
unsigned int number_of_divisors(int n){
    // ...
}
```

Beispiel

6 hat 4 Divisoren, nämlich 1, 2, 3, 6

Anzahl Divisoren (Lösung)

```
// PRE:  0 < n < MAX_INT
// POST: returns number of divisors of n (incl. 1 and n)
unsigned int number_of_divisors(int n){

    assert(n > 0);
    unsigned int counter = 0;

    for (int i = 1; i <= n; ++i){
        if(n % i == 0){
            counter++;
        }
    }

    return counter;
}
```

Fragen/Unklarheiten?

10. Exam Question

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

Fragen/Unklarheiten?

11. Headers und Namespaces

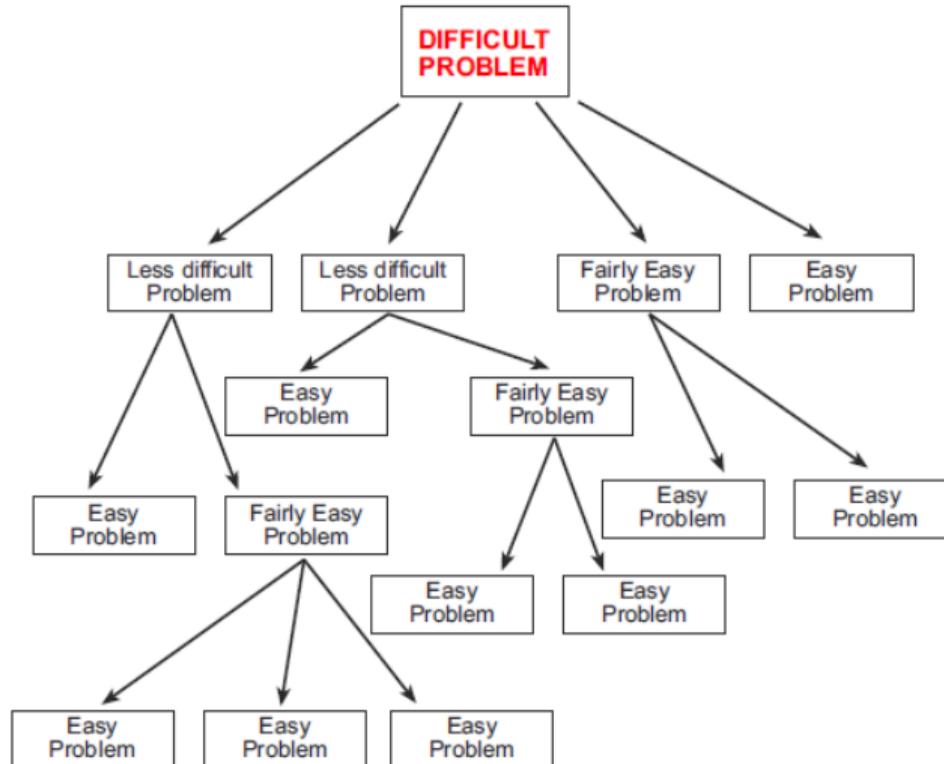
Headers und Namespaces

Live Demo Code Example "C++ Headers & Namespaces"

Fragen/Unklarheiten?

12. Stepwise Refinement

Grundidee



Stepwise Refinement

Code Example "Perfect Numbers" on `code expert`

Write a program that counts how many perfect numbers exist in the range $[a, b]$. Please use stepwise refinement to develop a solution to this task that is divided into meaningful functions. We provide a function `is_perfect` in `perfect.h` that checks if a given number is perfect.

A number $n \in \mathbb{N}$ is called perfect if and only if it is equal to the sum of its proper divisors. For example:

- $28 = 1 + 2 + 4 + 7 + 14$ is perfect
- $12 \neq 1 + 2 + 3 + 4 + 6$ is not perfect

Stepwise Refinement

- *nicht sofort programmieren!*
- identifiziert die einfacheren Teilprobleme

Stepwise Refinement

- *nicht sofort programmieren!*
- identifiziert die einfacheren Teilprobleme
- welche Teilprobleme konntet ihr identifizieren?

"Problembaum"

Wie viele vollkommene Zahlen (engl. perfect numbers) gibt es in $[a, b]$?

Lösung zu "Perfect Numbers"

```
// PRE:  
// POST:  
bool is_perfect(unsigned int number) {  
    unsigned int sum = 0;  
    for (unsigned int d = 1; d < number; ++d) {  
        if (number % d == 0) {  
            sum += d;  
        }  
    }  
    return sum == number;  
}
```

Lösung zu "Perfect Numbers"

```
#include <iostream>
#include "perfect.h"

// PRE:
// POST:
unsigned int count_perfect_numbers(unsigned int a, unsigned int b) {
    unsigned int count = 0;
    for (unsigned int i = a; i <= b; ++i) {
        if (is_perfect(i)) {
            count++;
        }
    }
    return count;
}

// ...
```

Lösung zu "Perfect Numbers"

```
// ...  
  
int main () {  
    // input  
    unsigned int a;  
    unsigned int b;  
    std::cin >> a >> b;  
  
    // computation  
    unsigned int count = count_perfect_numbers(a, b);  
  
    // output  
    std::cout << count << std::endl;  
  
    return 0;  
}
```

Fragen/Unklarheiten?

13. Alte Prüfungsfragen

Prüfungsfrage "Typ und Wert"

Geben Sie den Typ und den Wert der Variablen `c` an⁷

⁷Bemerkung zu Typ- und Wertfragen: Das Keyword **auto** bedeutet, dass der Typ des Ausdrucks durch den Compiler bestimmt wird. Im Folgenden steht es also für den Typ des Ausdrucks, die Sie identifizieren müssen.

Prüfungsfrage "Typ und Wert"

Geben Sie den Typ und den Wert der Variablen c an⁷

```
int a = 5;  
int b = 1;  
auto c = (9 * a + b) % a;
```

```
int a = 5;  
double b = 1;  
auto c = (9.0 * a + b) / a;
```

⁷Bemerkung zu Typ- und Wertfragen: Das Keyword **auto** bedeutet, dass der Typ des Ausdrucks durch den Compiler bestimmt wird. Im Folgenden steht es also für den Typ des Ausdrucks, die Sie identifizieren müssen.

Prüfungsfrage "Typ und Wert"

Geben Sie den Typ und den Wert der Variablen `c` an⁷

```
int a = 5;  
int b = 1;  
auto c = (9 * a + b) % a;
```

```
int a = 5;  
double b = 1;  
auto c = (9.0 * a + b) / a;
```

Lösung

⁷Bemerkung zu Typ- und Wertfragen: Das Keyword `auto` bedeutet, dass der Typ des Ausdrucks durch den Compiler bestimmt wird. Im Folgenden steht es also für den Typ des Ausdrucks, die Sie identifizieren müssen.

Prüfungsfrage "Typ und Wert"

Geben Sie den Typ und den Wert der Variablen c an⁷

```
int a = 5;  
int b = 1;  
auto c = (9 * a + b) % a;
```

```
int a = 5;  
double b = 1;  
auto c = (9.0 * a + b) / a;
```

Lösung

`int`,

⁷Bemerkung zu Typ- und Wertfragen: Das Keyword `auto` bedeutet, dass der Typ des Ausdrucks durch den Compiler bestimmt wird. Im Folgenden steht es also für den Typ des Ausdrucks, die Sie identifizieren müssen.

Prüfungsfrage "Typ und Wert"

Geben Sie den Typ und den Wert der Variablen c an⁷

```
int a = 5;  
int b = 1;  
auto c = (9 * a + b) % a;
```

```
int a = 5;  
double b = 1;  
auto c = (9.0 * a + b) / a;
```

Lösung

`int, 1`

Lösung

⁷Bemerkung zu Typ- und Wertfragen: Das Keyword `auto` bedeutet, dass der Typ des Ausdrucks durch den Compiler bestimmt wird. Im Folgenden steht es also für den Typ des Ausdrucks, die Sie identifizieren müssen.

Prüfungsfrage "Typ und Wert"

Geben Sie den Typ und den Wert der Variablen c an⁷

```
int a = 5;  
int b = 1;  
auto c = (9 * a + b) % a;
```

```
int a = 5;  
double b = 1;  
auto c = (9.0 * a + b) / a;
```

Lösung

`int, 1`

Lösung

`double,`

⁷Bemerkung zu Typ- und Wertfragen: Das Keyword `auto` bedeutet, dass der Typ des Ausdrucks durch den Compiler bestimmt wird. Im Folgenden steht es also für den Typ des Ausdrucks, die Sie identifizieren müssen.

Prüfungsfrage "Typ und Wert"

Geben Sie den Typ und den Wert der Variablen c an⁷

```
int a = 5;  
int b = 1;  
auto c = (9 * a + b) % a;
```

```
int a = 5;  
double b = 1;  
auto c = (9.0 * a + b) / a;
```

Lösung

`int, 1`

Lösung

`double, 9.2`

⁷Bemerkung zu Typ- und Wertfragen: Das Keyword `auto` bedeutet, dass der Typ des Ausdrucks durch den Compiler bestimmt wird. Im Folgenden steht es also für den Typ des Ausdrucks, die Sie identifizieren müssen.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem⁸

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

Richtig oder Falsch?

⁸Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem⁸

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

Richtig oder Falsch?

1. "1.25 kann im Fließkommazahlensystem exakt dargestellt werden"

⁸Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem⁸

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

Richtig oder Falsch?

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

⁸Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem⁸

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

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: $0.0625 < Z < 0.25$ "

⁸Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem⁸

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

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., $1.0 * 2^{-1}$)

⁸Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem⁸

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

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"

⁸Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem⁸

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

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

⁸Erinnerung: 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!

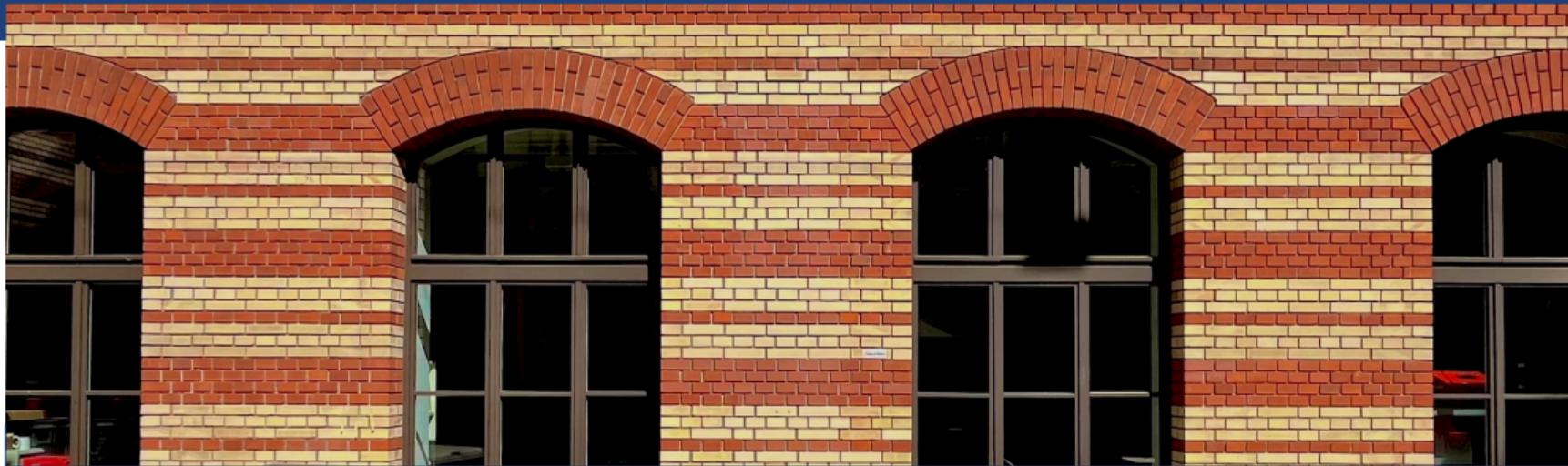
- Division von zwei positiven `int` kann nicht negativ sein
⇒ `sum >= 0` ist immer wahr!
- Nach der ersten Ausführung des do-Blocks: `i > sum`.
`sum` ist monoton fallend, `i` ist monoton steigend
⇒ `i > sum` ist immer wahr.

14. Outro

Allgemeine Fragen?

Bis zum nächsten Mal

Schöne Woche noch!



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;
```

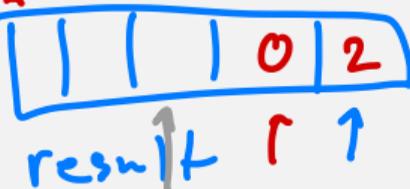
```
    }
```

```
    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?

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 "Cutoff-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 "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. $1.01100 \cdot 2^2$

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 "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. $1.01010 \cdot 2^2 \approx 1.010$

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)

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 "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. $1.01110 \cdot 2^2 \approx 1.\underline{100}$

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 "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. $1.010\underline{10} \cdot 2^2 \approx 1.010$ 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 "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. $1.01100 \cdot 2^2 \approx 1.011$

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 "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. $1.01100 \cdot 2^2 \approx 1.011$ 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?

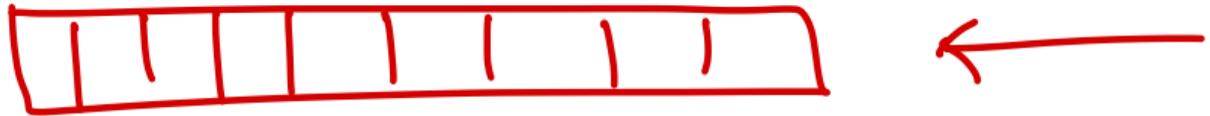
- `#include <vector>`

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?

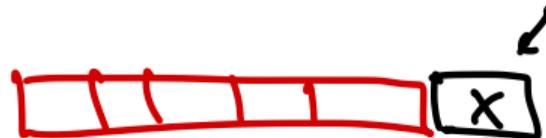
- `#include <vector>`
- Vektoren kann man sich als eine Reihe von Kästchen vorstellen, die je einen Wert des gegebenen Typs speichern
- Man kann Vektoren etwa so behandeln wie einen neuen Typen
- Es gibt viele Möglichkeiten, einen Vector zu initialisieren/definieren. Schaut im Vorlesungsmaterial nach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

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

$(0.5 + 0.5) + 10$	
decimal	binary
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		?????
=		$1.01 \cdot 2^3$	=		?????
+ 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		$1010.00 \cdot 2^0$
=		←	$1.01 \cdot 2^3$	=		←	$1011.00 \cdot 2^0$

$(10 + 0.5) + 0.5$		$(0.5 + 0.5) + 10$	
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	$\leftarrow 1.01 \cdot 2^3$	= ??	$\leftarrow 1.011 \cdot 2^3$

$(10 + 0.5) + 0.5$		$(0.5 + 0.5) + 10$	
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!

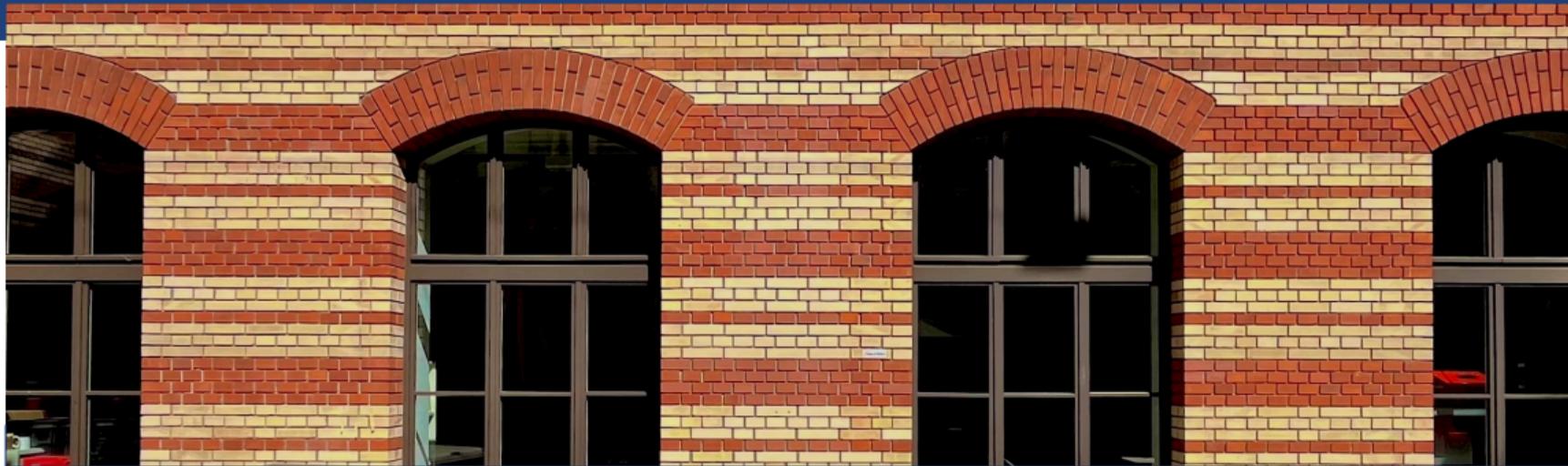
- Division von zwei positiven `int` kann nicht negativ sein
⇒ `sum >= 0` ist immer wahr!
- Nach der ersten Ausführung des do-Blocks: `i > sum`.
`sum` ist monoton fallend, `i` ist monoton steigend
⇒ `i > sum` ist immer wahr.

10. Outro

Allgemeine Fragen?

Bis zum nächsten Mal

Schöne Woche noch!



Übungsstunde — Informatik — 07

Adel Gavranović

Follow-up, ASCII-Zeichen, `char`, Rekursion, viel **code expert**

Übersicht

Follow-up

`const` und Funktionen

Repetition: Fließkommazahlen

(ASCII) Zeichen in C++ (`char`)

Rekursion

Feedback

Alte Prüfungsfragen



`n.ethz.ch/~agavranovic`

 Material

 Webpage

 Mail

1. Follow-up

Follow-up aus letzter Übungsstunde

Follow-up aus letzter Übungsstunde

- Nicht viel Follow-up heute

Follow-up aus letzter Übungsstunde

- Nicht viel Follow-up heute
- Habe ich etwas vergessen?
- Werde mir in Zukunft Fragen besser notieren...

1. Follow-up

1.1. const und Funktionen

const und Funktionen

const und Funktionen

```
1. | void Funktion(const Type& n){           // const als Arugment  
    |     // n kann nicht verändert werden  
    | } // sehr oft sinnvoll; muss man kennen!
```

const und Funktionen

1.

```
void Funktion(const Type& n){           // const als Argument
    // n kann nicht verändert werden
} // sehr oft sinnvoll; muss man kennen!
```
2.

```
const Type& Funktion(const Type& n){    // const als Return
    // n kann nicht verändert werden
    return n; // ursprüngliches n wird returned
} // manchmal sinnvoll; ziemlich spezifisch
```

const und Funktionen

1.

```
void Funktion(const Type& n){           // const als Argument
    // n kann nicht verändert werden
} // sehr oft sinnvoll; muss man kennen!
```

2.

```
const Type& Funktion(const Type& n){    // const als Return
    // n kann nicht verändert werden
    return n; // ursprüngliches n wird returned
} // manchmal sinnvoll; ziemlich spezifisch
```

3.

```
void Funktion(Type n) const {           // const-Funktionen
    // n kann verändert werden
    // nichts in der Klasse, der die Funktion angehört,
    // kann verändert werden. [Klassen wurden noch nicht behandelt]
} // oft sinnvoll; wird vielleicht besprochen im Thema "Klassen"
```

Fragen/Unklarheiten?

1. Follow-up

1.2. 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	$\leftarrow 1.01 \cdot 2^3$	= ??	\leftarrow ?????

$(10 + 0.5) + 0.5$		$(0.5 + 0.5) + 10$	
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	$\leftarrow 1.01 \cdot 2^3$	= ??	$\leftarrow 1.011 \cdot 2^3$

$(10 + 0.5) + 0.5$		$(0.5 + 0.5) + 10$	
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?

2. Feedback zu **code** expert

Fragen bezüglich **code expert** eurerseits?

3. Lernziele

Ziele für Heute

Lernziele

Ziele für Heute

Lernziele

- Programme, die `char` verwenden, tracen und schreiben können
- Programme, die Rekursion verwenden, tracen und schreiben können

4. Zusammenfassung

5. (ASCII) Zeichen in C++ (char)

Übung "Converting Input to UPPER CASE"

Aufgabe

1. Überlegt, wie ihr die Aufgabe "Converting Input to UPPER CASE" auf **code expert** am besten angeht

Übung "Converting Input to UPPER CASE"

Aufgabe

1. Überlegt, wie ihr die Aufgabe "Converting Input to UPPER CASE" auf **code expert** am besten angeht
2. Programmiert (optional gruppenweise) eine Lösung

Übung "Converting Input to UPPER CASE"

Aufgabe

Schreibt ein Programm, das eine durch ein Zeilenumbruchszeichen begrenzte Zeichenfolge als Vektor von `char` einliest. Anschließend soll das Programm die Folge ausgeben, wobei alle Kleinbuchstaben in GROSSBUCHSTABEN umgewandelt werden.

"Converting Input to UPPER CASE" — Lösung

```
#include <iostream>
#include <vector>
#include <ios>           // not really needed, don't worry about it
```

"Converting Input to UPPER CASE" — Lösung

```
// POST: Converts the letter to upper case.
void char_to_upper(char& letter){
    int shift_distance = 'a' - 'A';    // 'a' > 'A' (if conv. to ints)
                                        // distance between the upper
                                        // and lower case numbers

    if('a' <= letter && letter <= 'z'){
        letter -= shift_distance;
    }
}

// POST: Converts all letters to upper-case.
void to_upper(std::vector<char>& letters){
    for(int i = 0; i < letters.size(); ++i){
        char_to_upper(letters.at(i));
    }
}
```

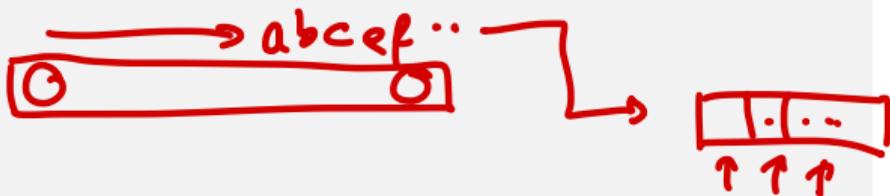
"Converting Input to UPPER CASE" — Lösung

```
std::vector<char> letters;
char ch;

// Step 1: Read input.
do {
    std::cin >> ch;
    letters.push_back(ch);
} while(ch != '\n');

// Step 2: Convert to upper-case.
to_upper(letters);

// Step 3: Output.
for(int i = 0; i < letters.size(); ++i){
    std::cout << letters.at(i);
}
```



Fragen/Unklarheiten?

6. Rekursion

Alte Prüfungsaufgabe

Eckdaten

- Prüfung: 01.2022 Computer Science (MATH/PHYS/RW)
- Einfache Rekursionsaufgabe
- Gesamtpunkte in der Prüfung: 85 Punkte
- Gesamtzeit für die Prüfung: 120 minutes
- Punkte für die Aufgabe: 5 Punkte

Alte Prüfungsaufgabe

Eckdaten

- Prüfung: 01.2022 Computer Science (MATH/PHYS/RW)
- Einfache Rekursionsaufgabe
- Gesamtpunkte in der Prüfung: 85 Punkte
- Gesamtzeit für die Prüfung: 120 minutes
- Punkte für die Aufgabe: 5 Punkte
- Geschätzte Zeit für die Aufgabe: 7 minutes = $120 * (5/85)$

Kleine Prüfungssimulation

- Schalten in den "Prüfungsmodus" und stellt alles was ihr brauchen könnten bereit

Kleine Prüfungssimulation

- Schalten in den "Prüfungsmodus" und stellt alles was ihr brauchen könnten bereit
- Öffnet "[Exam 2022.01 (MATH/PHYS/RW)] Compute Series" auf **code expert**

Kleine Prüfungssimulation

- Schalten in den "Prüfungsmodus" und stellt alles was ihr brauchen könnten bereit
- Öffnet "[Exam 2022.01 (MATH/PHYS/RW)] Compute Series" auf **code expert**
- Programmiert eine (rekursive) Lösung
- Zeit: 7 Minuten

Alte Prüfungsaufgabe

Wir wollen eine Funktion mit den folgenden PRE und POSTs schreiben

```
// PRE:   a positive integer n
//
// POST:  returns the n-th number of a series x_n, defined as
//        x_n = 2,           for n = 1
//        x_n = 1,           for n = 2 } Base case
//        x_n = x_(n-1) + x_(n-2), for n > 2 } recursive
//
// Example:
// * n == 1 ~~> 2
// * n == 2 ~~> 1
// * n == 3 ~~> 3
```

Alte Prüfungsaufgabe – Lösung

Alte Prüfungsaufgabe — Lösung

```
// PRE:   a positive integer n
//
// POST:  returns the n-th number of a serie x_n, defined as
//        x_n = 2,           for n = 1
//        x_n = 1,           for n = 2
//        x_n = x_(n-1) + x_(n-2),   for n > 2

int compute_element(int n) {
    if (n == 1) {
        return 2;
    } else if (n == 2) {
        return 1;
    } else {
        return compute_element(n-1) + compute_element(n-2);
    }
}
```

Fragen/Unklarheiten?

Übung "Partial Sum"

Aufgabe

Schreibe eine Funktion, die

1. die Summe aller natürlichen Zahlen kleiner oder gleich n per Rekursion berechnet und diesen Wert zurückgibt
2. alle addierten Terme in aufsteigender Reihenfolge (von 0 bis n) in der gleichen Rekursiven Funktion ausdrückt

↳ wird tricky!

es gibt iterative
Lösungen, aber wir
üben jetzt Rekursion!
↑

Übung "Partial Sum"

- Öffnet "Partial Sum" auf **code expert**

Übung "Partial Sum"

- Öffnet "Partial Sum" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet

Übung "Partial Sum"

- Öffnet "Partial Sum" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet
- Programmiert eine (rekursive) Lösung (optional in Gruppen)

"Partial Sum" — Lösung

"Partial Sum" — Lösung

```
int partial_sum(const int n) {
    if (n == 0) {
        return 0;
    } else {
        // print descending
        // std::cout << n << std::endl;

        int partial = partial_sum(n - 1);

        // print ascending
        std::cout << n << std::endl;

        return n + partial;
    }
}
```

□
[(0, n)
↑
?

"Partial Sum" — Lösung

```
int main() {  
    std::cout << "n = ";  
  
    int n;  
    std::cin >> n;  
  
    std::cout << partial_sum(n) << std::endl;  
  
    return 0;  
}
```

Fragen/Unklarheiten?

Übung "Power Function"

Frage

Wie viele Rekursionsaufrufe braucht die folgende Funktion, um x^7 zu berechnen?

```
int power(const int x, const int n) {  
  
    if (n == 0){  
        return 1;  
    } else if (n == 1) {  
        return x;  
    }  
  
    return x * power(x, n - 1);  
}
```

Antwort:

Übung "Power Function"

Frage

→ Anfr. aus d. Funktion selbst



Wie viele Rekursionsaufrufe braucht die folgende Funktion, um x^7 zu berechnen?

```
int power(const int x, const int n) {  
  
    if (n == 0){  
        return 1;  
    } else if (n == 1) {  
        return x;  
    }  
  
    return x * power(x, n - 1);  
}
```

Antwort: 7

Übung "Power Function"

- Öffnet "Power Function" auf **code expert**

Übung "Power Function"

- Öffnet "Power Function" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet

Übung "Power Function"

- Öffnet "Power Function" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet
- Programmiert eine (rekursive) Lösung
- *Tipp: Die Aufgabe ist eine Verallgemeinerung der Aufgabe "Multiply with 29" der ersten Woche*

Tipp: $a^{2n} = a^n \cdot a^n$



same value!

Tipp: $m \% 2 == 0 \Leftrightarrow \exists n: m = 2n$
(Don't show Ueli this)

"Power Function" — Lösung

```
// POST: result == x^n
int power (const int x, const int n) {
    if(n == 0) {
        return 1;
    } else if(n == 1) {
        return x;
    } else if(n % 2 == 0) {           // case n = 2m for some m in N
        int temp = power(x, n/2);   // temp, to not call the function twice!
        return temp * temp;        // since x^n = x^(2m) = x^m * x^m
    } else { n ungerade
        return x * power(x, n-1);
    }
    int temp = pow(x, n/2)
}
```

$$\begin{aligned}x^7 &= x \cdot x^6 \\ &= x \cdot (x^3)^2\end{aligned}$$

Fragen/Unklarheiten?

Die Türme von Hanoi

Mit dieser Übung zu kämpfen, ist ein lang gehegte Tradition unter Programmieranfänger:innen.

Sie ist berüchtigt schwierig, wenn man mit Rekursion nicht vertraut ist.

Die Türme von Hanoi

Mit dieser Übung zu kämpfen, ist ein lang gehegte Tradition unter Programmieranfänger:innen.

Sie ist berüchtigt schwierig, wenn man mit Rekursion nicht vertraut ist.

Everyone: it's a game for kids



Programmers:



Experiment: Die Türme von Hanoi



Links

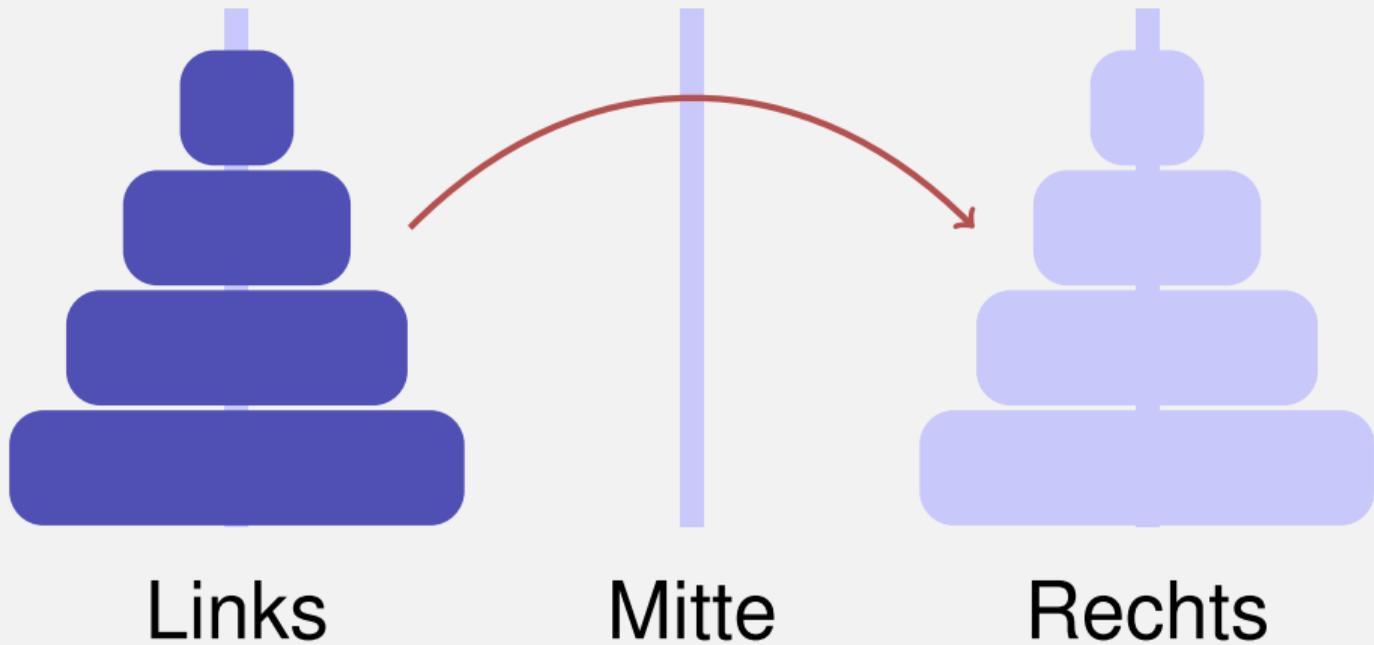


Mitte



Rechts

Experiment: Die Türme von Hanoi



Die Türme von Hanoi - So gehts!



Links



Mitte



Rechts

Die Türme von Hanoi - So gehts!



Links



Mitte



Rechts

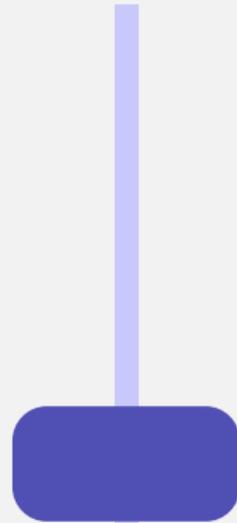
Die Türme von Hanoi - So gehts!



Links



Mitte



Rechts

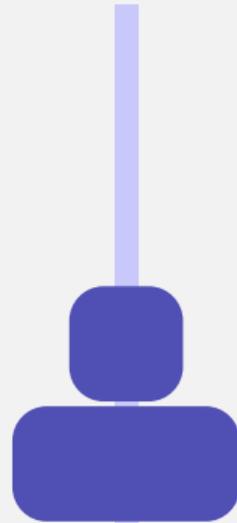
Die Türme von Hanoi - So gehts!



Links

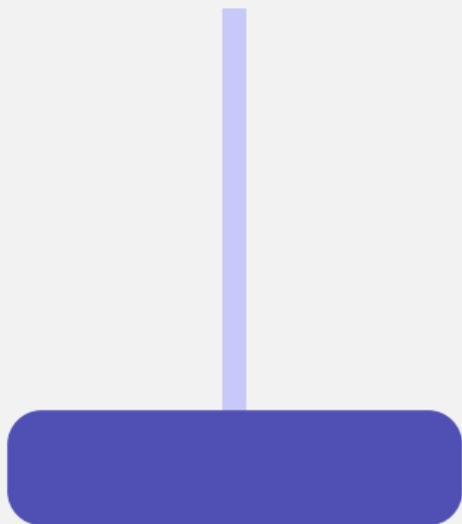


Mitte

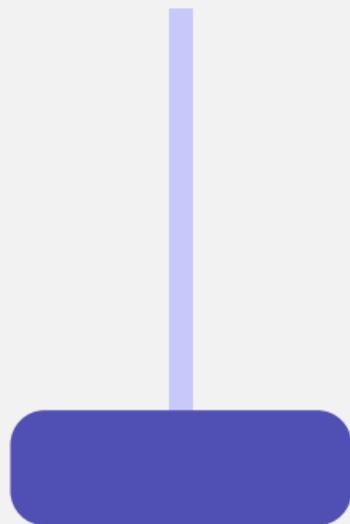


Rechts

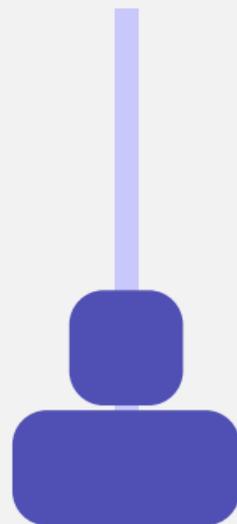
Die Türme von Hanoi - So gehts!



Links

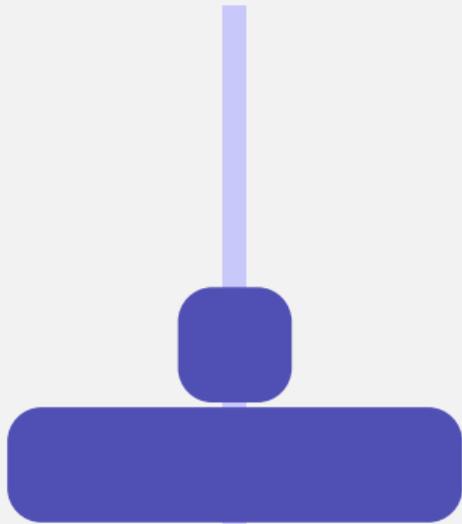


Mitte

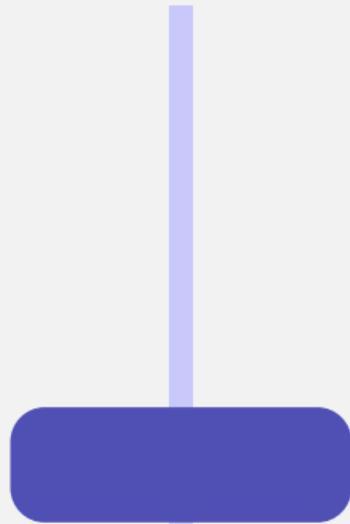


Rechts

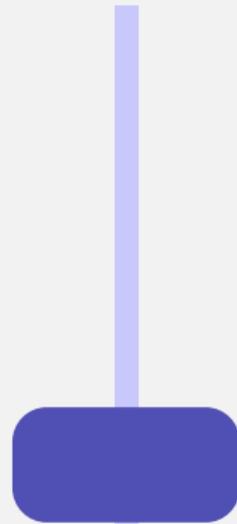
Die Türme von Hanoi - So gehts!



Links

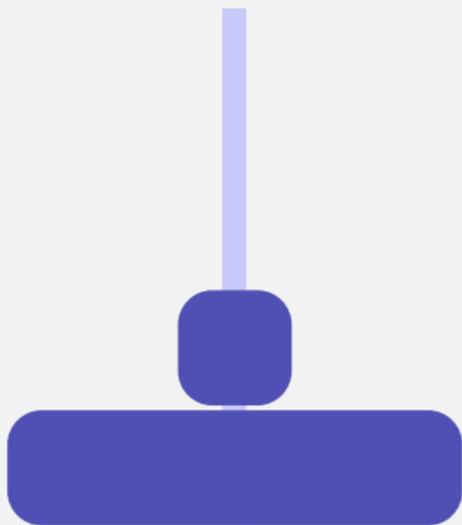


Mitte

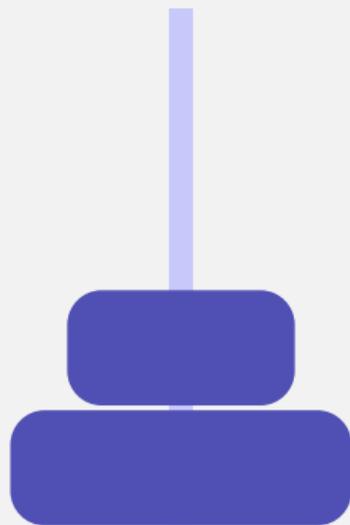


Rechts

Die Türme von Hanoi - So gehts!



Links

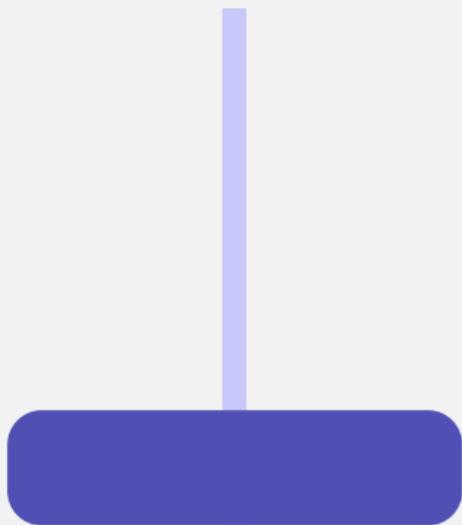


Mitte

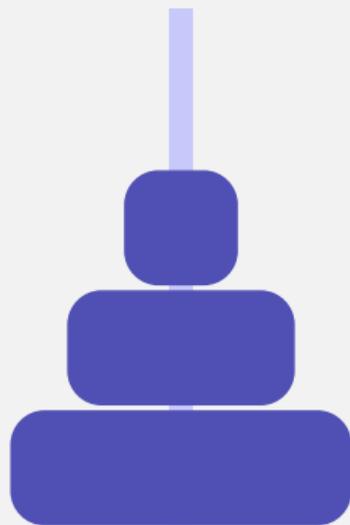


Rechts

Die Türme von Hanoi - So gehts!



Links

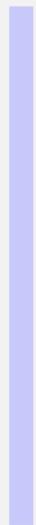


Mitte

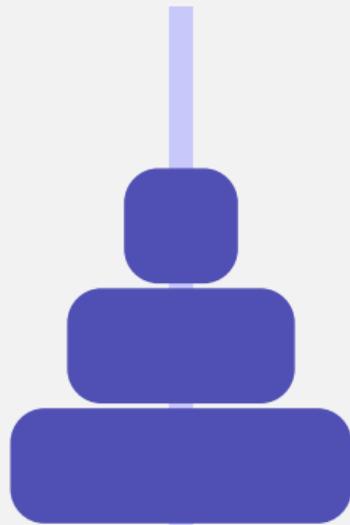


Rechts

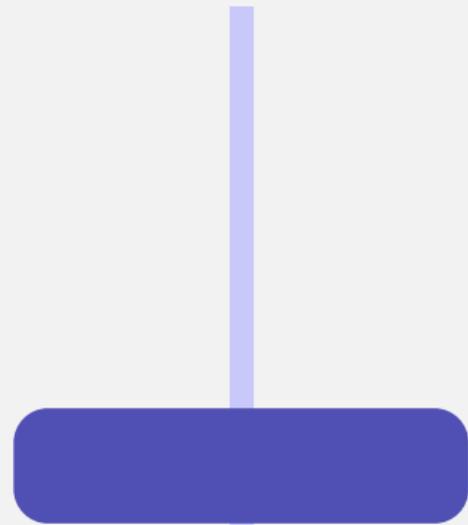
Die Türme von Hanoi - So gehts!



Links



Mitte

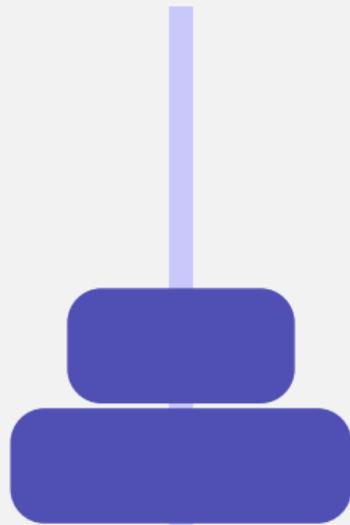


Rechts

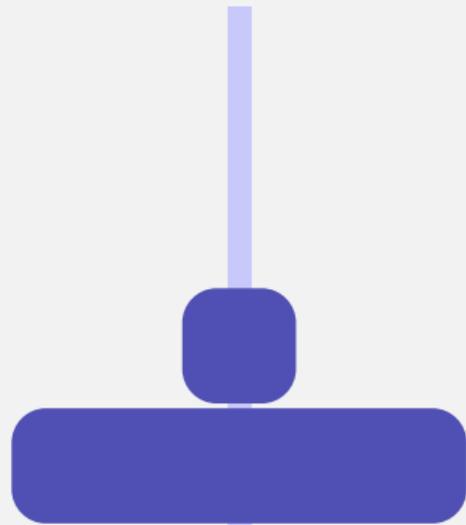
Die Türme von Hanoi - So gehts!



Links

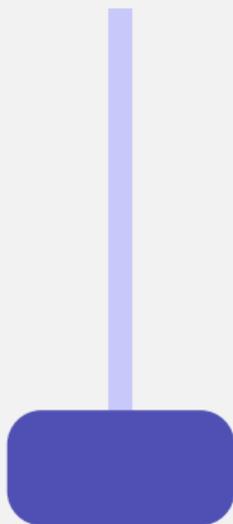


Mitte

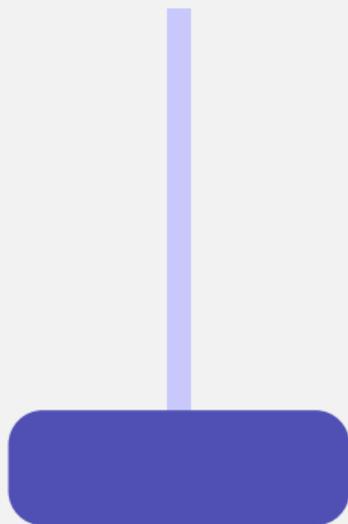


Rechts

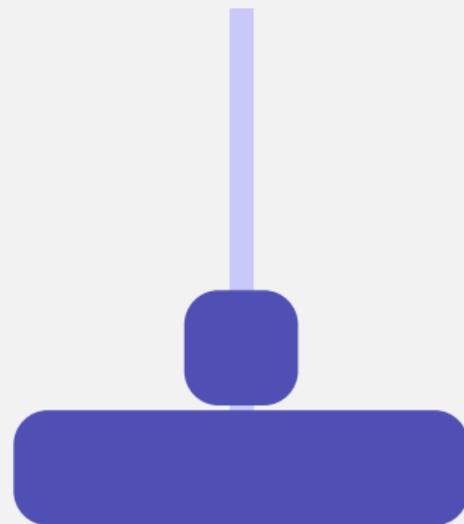
Die Türme von Hanoi - So gehts!



Links

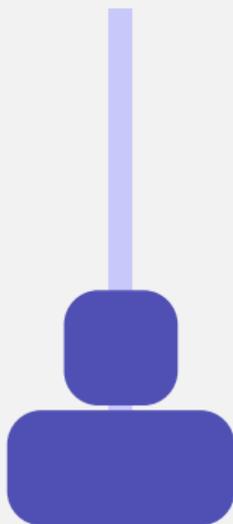


Mitte

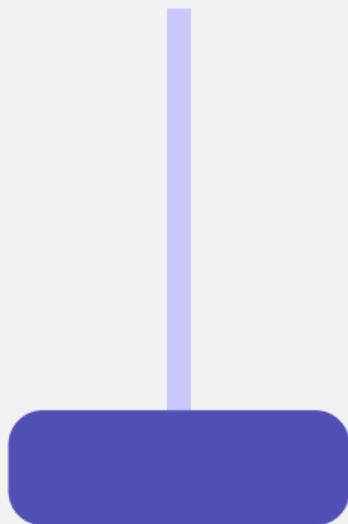


Rechts

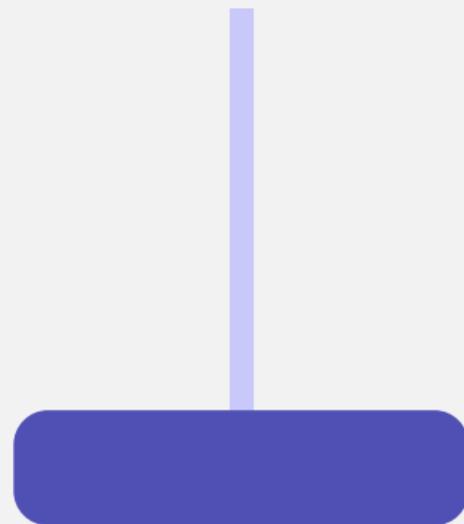
Die Türme von Hanoi - So gehts!



Links

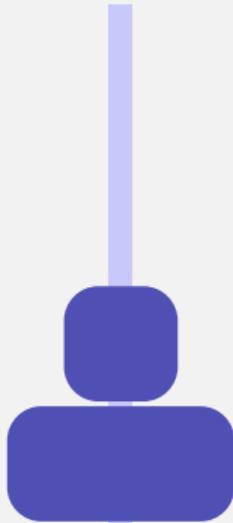


Mitte



Rechts

Die Türme von Hanoi - So gehts!



Links

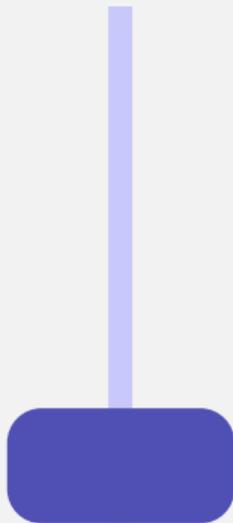


Mitte



Rechts

Die Türme von Hanoi - So gehts!



Links

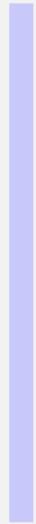


Mitte

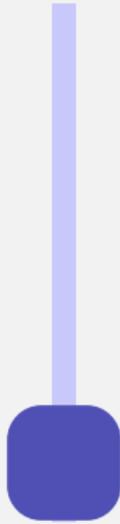


Rechts

Die Türme von Hanoi - So gehts!



Links



Mitte



Rechts

Die Türme von Hanoi - So gehts!



Links



Mitte



Rechts

Die Türme von Hanoi - Rekursiver Lösungsansatz



Links

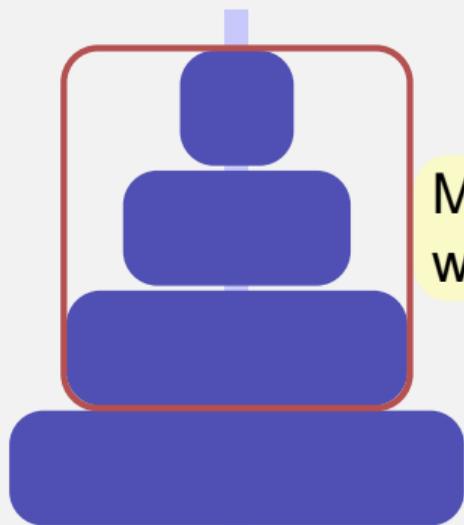


Mitte



Rechts

Die Türme von Hanoi - Rekursiver Lösungsansatz



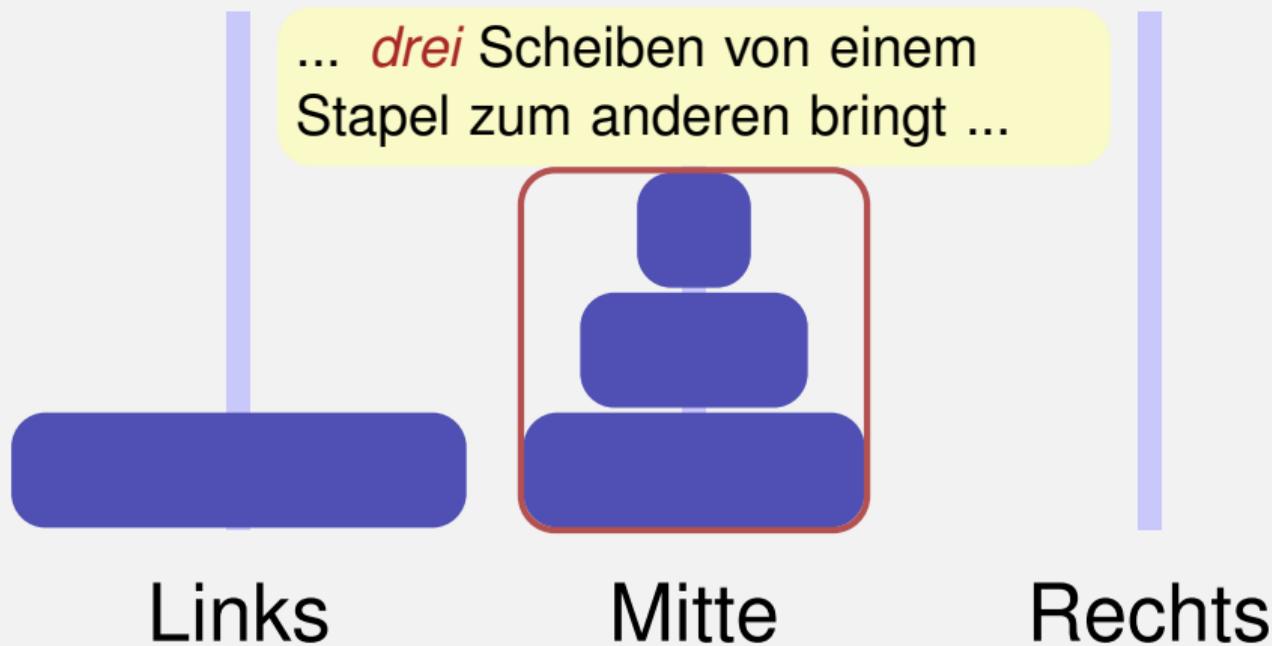
Links

Mal *angenommen*, wir
wüssten wie man ...

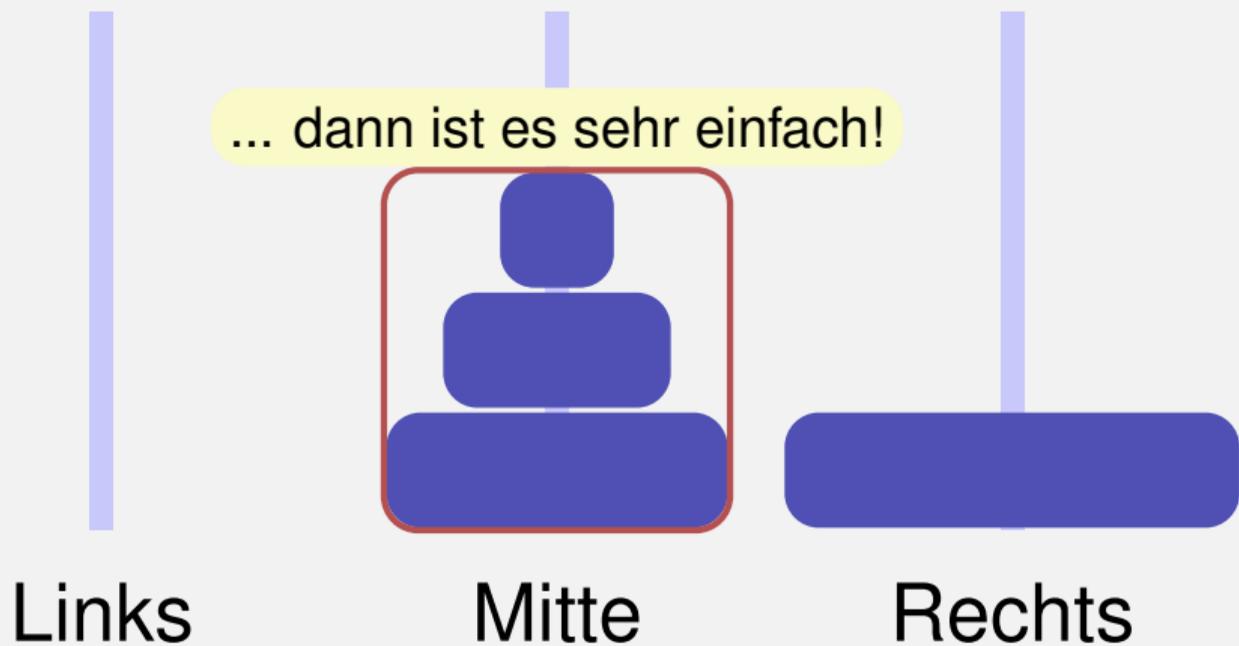
Mitte

Rechts

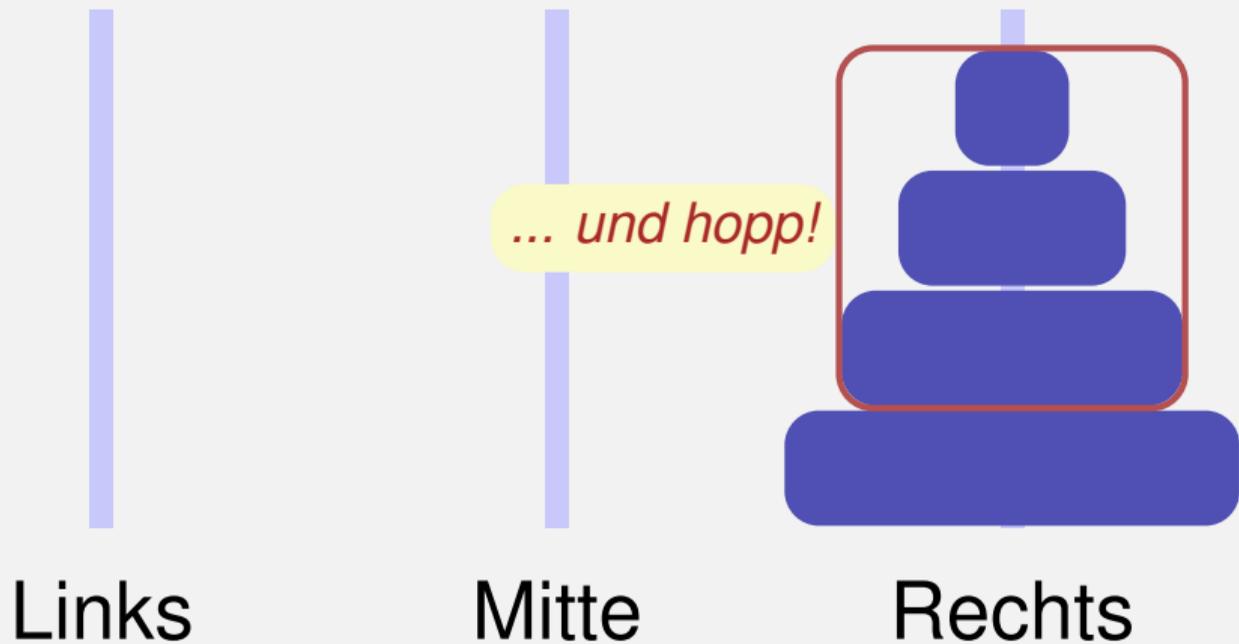
Die Türme von Hanoi - Rekursiver Lösungsansatz



Die Türme von Hanoi - Rekursiver Lösungsansatz



Die Türme von Hanoi - Rekursiver Lösungsansatz



Die Türme von Hanoi - Rekursiver Lösungsansatz



Links

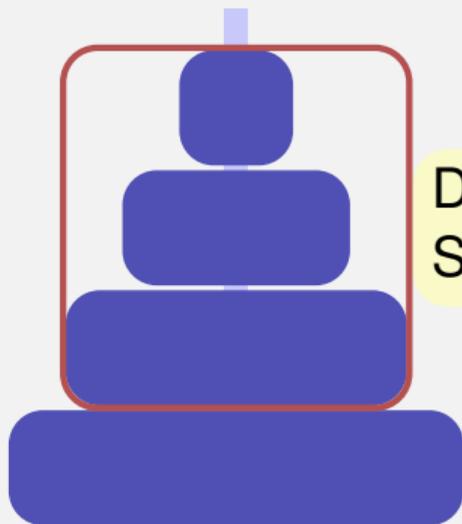


Mitte



Rechts

Die Türme von Hanoi - Rekursiver Lösungsansatz



Links

Doch *wie* können wir drei
Scheiben bewegen?

Mitte

Rechts

Die Türme von Hanoi - Rekursiver Lösungsansatz



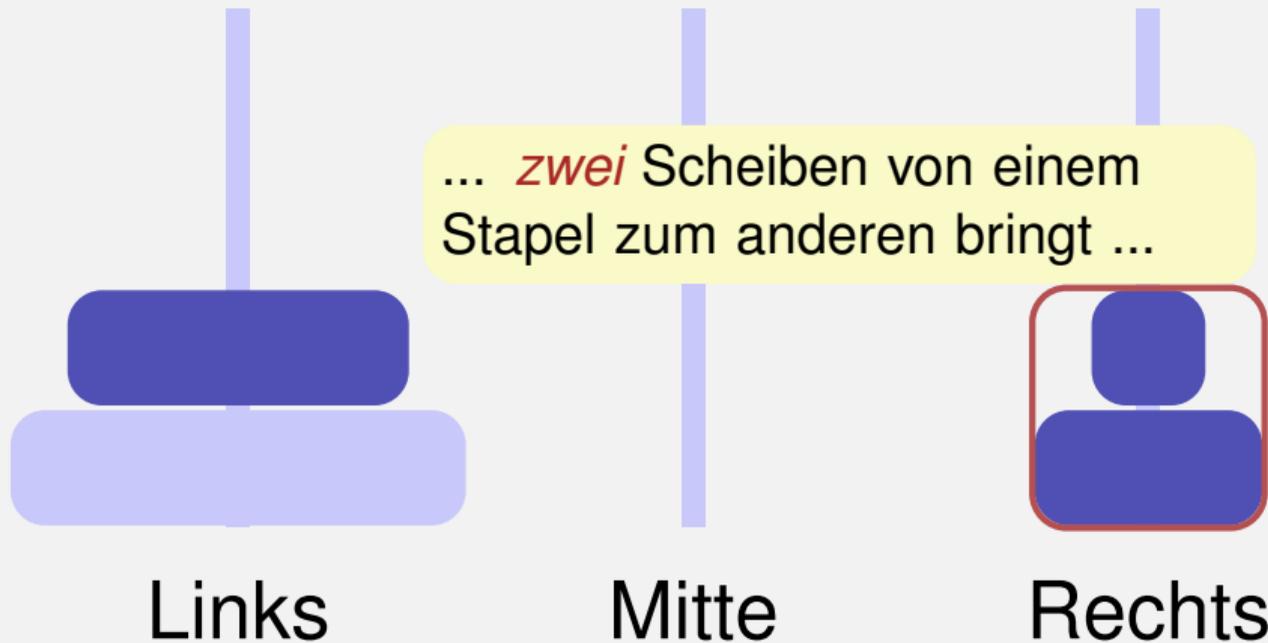
Mal *angenommen*, wir wüssten wie man ...

Links

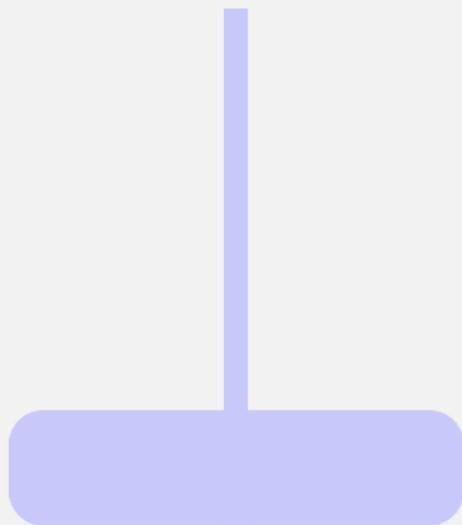
Mitte

Rechts

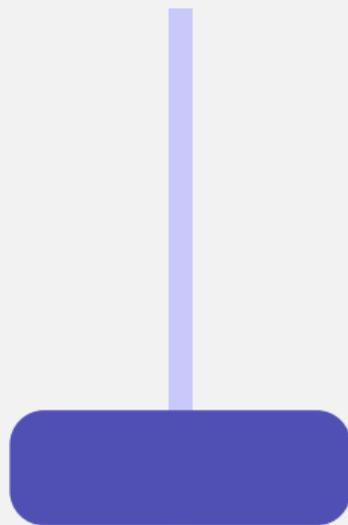
Die Türme von Hanoi - Rekursiver Lösungsansatz



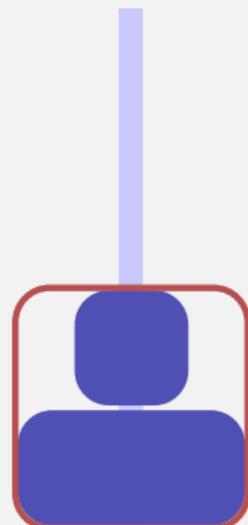
Die Türme von Hanoi - Rekursiver Lösungsansatz



Links

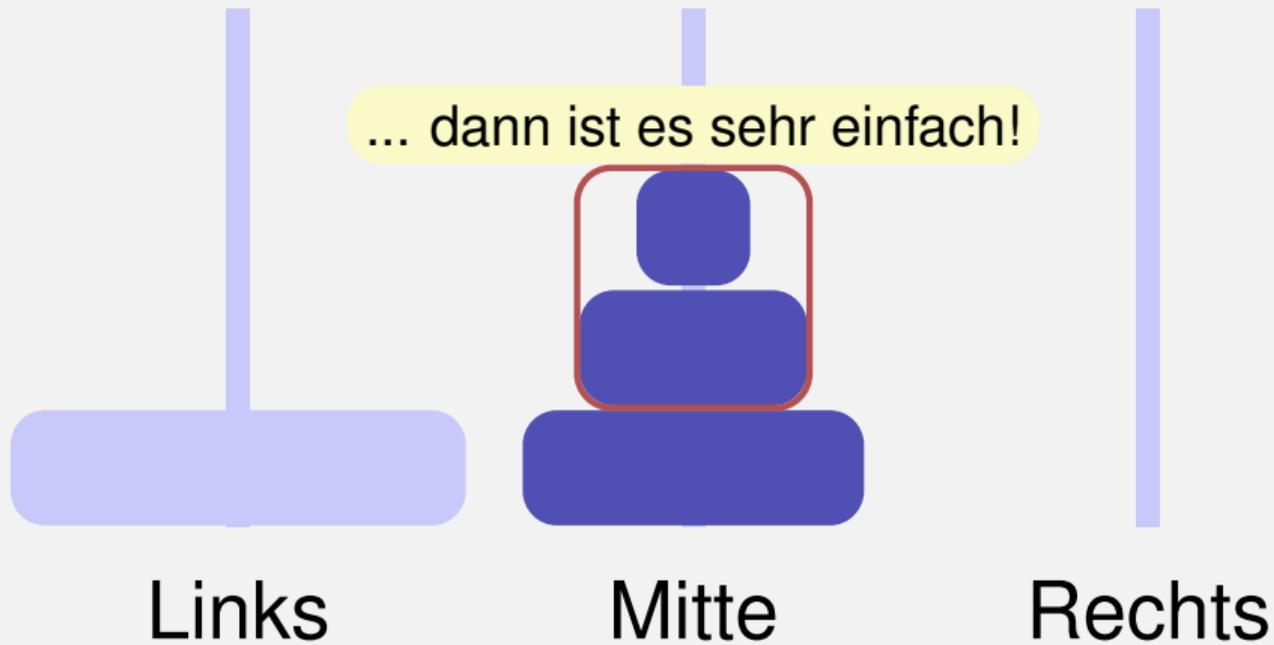


Mitte

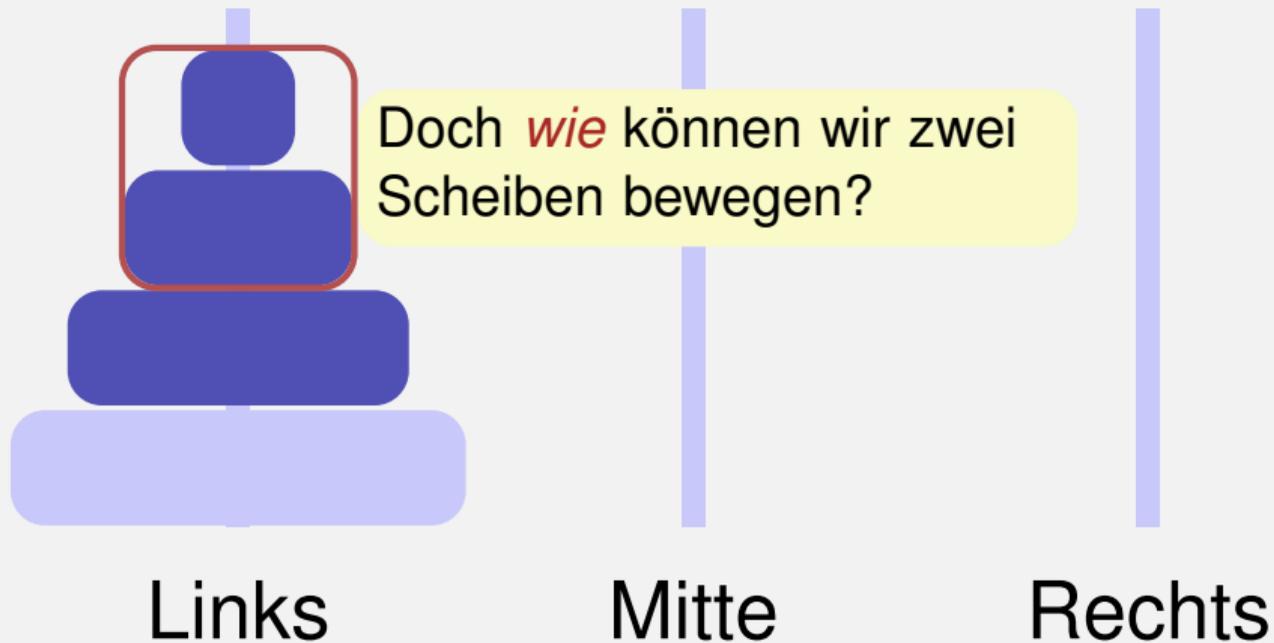


Rechts

Die Türme von Hanoi - Rekursiver Lösungsansatz



Die Türme von Hanoi - Rekursiver Lösungsansatz



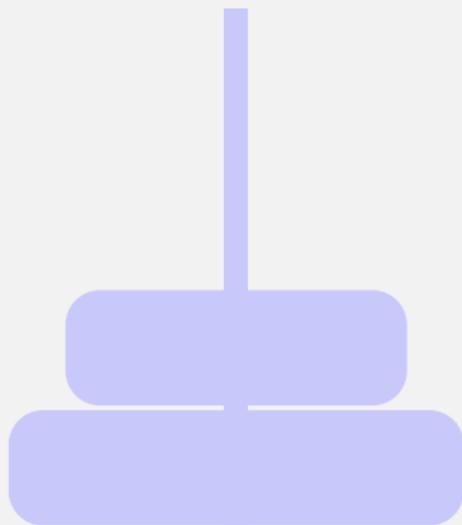
Die Türme von Hanoi - Rekursiver Lösungsansatz



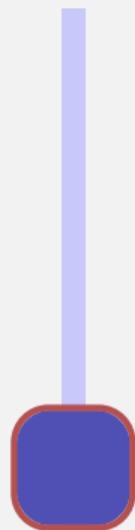
Die Türme von Hanoi - Rekursiver Lösungsansatz



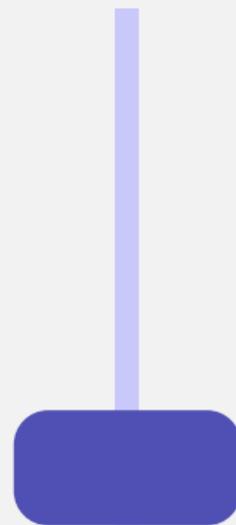
Die Türme von Hanoi - Rekursiver Lösungsansatz



Links



Mitte



Rechts

Die Türme von Hanoi - Rekursiver Lösungsansatz



Aufgabe Towers of Hanoi

- Öffnet "Towers of Hanoi" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet
- Programmiert eine Lösung (optional in Gruppen)

Die Türme von Hanoi - Code



left

middle

right

Bewege 4 Scheiben von left nach right mit Hilfsstapel middle:

```
move(4, "left", "middle", "right")
```

Die Türme von Hanoi - Code

`move(n, src, aux, dst)` \Rightarrow

- 1 Bewege die obersten $n - 1$ Scheiben von *src* nach *aux* mit Hilfsstapel *dst*:
`move(n - 1, src, dst, aux);`
- 2 Bewege 1 Scheibe von *src* nach *dst*
`move(1, src, aux, dst);`
- 3 Bewege die obersten $n - 1$ Scheiben von *aux* nach *dst* mit Hilfsstapel *src*:
`move(n - 1, aux, src, dst);`

Die Türme von Hanoi - Code

```
void move(int n, const string &src, const string &aux, const string &dst){
    if (n == 1) {
        // base case ('move' the disc)
        std::cout << src << " --> " << dst << std::endl;
    } else {
        // recursive case

    }
}
```

Die Türme von Hanoi - Code

```
void move(int n, const string &src, const string &aux, const string &dst){  
    if (n == 1) {  
        // base case ('move' the disc)  
        std::cout << src << " --> " << dst << std::endl;  
    } else {  
        // recursive case  
        move(n-1, src, dst, aux);  
  
    }  
}
```

Die Türme von Hanoi - Code

```
void move(int n, const string &src, const string &aux, const string &dst){
    if (n == 1) {
        // base case ('move' the disc)
        std::cout << src << " --> " << dst << std::endl;
    } else {
        // recursive case
        move(n-1, src, dst, aux);
        move(1, src, aux, dst);
    }
}
```

Die Türme von Hanoi - Code

```
void move(int n, const string &src, const string &aux, const string &dst){  
    if (n == 1) {  
        // base case ('move' the disc)  
        std::cout << src << " --> " << dst << std::endl;  
    } else {  
        // recursive case  
        move(n-1, src, dst, aux);  
        move(1, src, aux, dst);  
        move(n-1, aux, src, dst);  
    }  
}
```

Die Türme von Hanoi - Code

```
void move(int n, const string &src, const string &aux, const string &dst){
    if (n == 1) {
        // base case ('move' the disc)
        std::cout << src << " --> " << dst << std::endl;
    } else {
        // recursive case
        move(n-1, src, dst, aux);
        move(1, src, aux, dst);
        move(n-1, aux, src, dst);
    }
}

int main() {
    move(4, "left", "middle", "right");
    return 0;
}
```

Die Türme von Hanoi - Code Alternative

```
void move(int n, const string &src, const string &aux, const string &dst){  
    // base case  
    if (n == 0) return;  
  
    // recursive case  
    move(n-1, src, dst, aux);  
    std::cout << src << " --> " << dst << "\n";  
    move(n-1, aux, src, dst);  
}
```

```
int main() {  
    move(4, "left ", "middle", "right ");  
    return 0;  
}
```

Fragen/Unklarheiten?

Videoempfehlungen

Versucht insbesondere das Konzept von *Recursive Leap of Faith* nachzuvollziehen. Das ist quasi die Induktionsannahme bei einem Induktionsbeweis aus der Mathematik

Videos zum Thema Rekursion

- [5 Simple Steps for Solving Any Recursive Problem](#)
- [Towers of Hanoi: A Complete Recursive Visualization](#)

7. Feedback

Euer Feedback an mich

□ include more exam-like tasks



 Feedback-Formular

(Nehmt euch Zeit und seid ehrlich)

8. Alte Prüfungsfragen

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem¹

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

Richtig oder Falsch?

¹Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem¹

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

Richtig oder Falsch?

1. "1.25 kann im Fließkommazahlensystem exakt dargestellt werden"

¹Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem¹

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

Richtig oder Falsch?

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

¹Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem¹

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

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: $0.0625 < Z < 0.25$ "

¹Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem¹

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

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., $1.0 * 2^{-1}$)

¹Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem¹

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

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"

¹Erinnerung: die Genauigkeit (Anzahl der Ziffern) schliesst das führende Bit ein.

Prüfungsfrage F^*

Sei F^* das folgende normalisierte Fließkommazahlensystem¹

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

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

¹Erinnerung: 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!

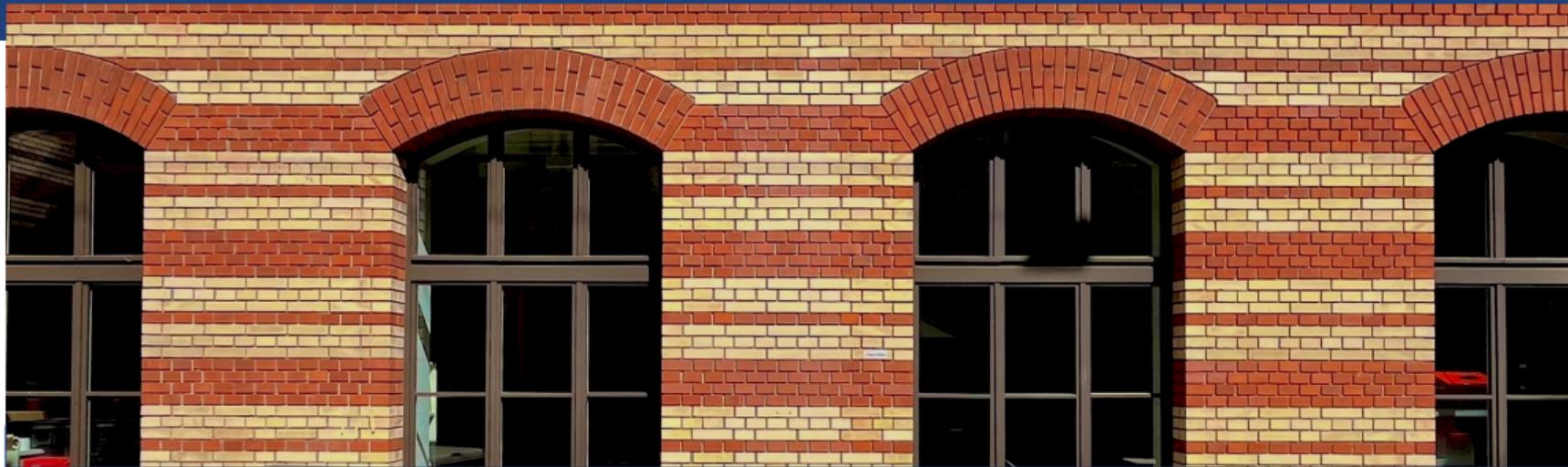
- Division von zwei positiven `int` kann nicht negativ sein
⇒ `sum >= 0` ist immer wahr!
- Nach der ersten Ausführung des do-Blocks: `i > sum`.
`sum` ist monoton fallend, `i` ist monoton steigend
⇒ `i > sum` ist immer wahr.

9. Outro

Allgemeine Fragen?

Bis zum nächsten Mal

Schöne Woche noch!



Übungsstunde — Informatik — 08

Adel Gavranović

Rekursion, Structs

Übersicht

Follow-up

`const` und Funktionen

Aufgabe "Towers of Hanoi"

Structs

Aufgabe "Geometry Exercise"

Rekursion

Aufgabe "Power Set"



`n.ethz.ch/~agavranovic`

 Material

 Webpage

 Mail

1. Follow-up

Follow-up aus letzter Übungsstunde

Follow-up aus letzter Übungsstunde

- Applausreform

Follow-up aus letzter Übungsstunde

- Applausreform
- Gute Nachrichten (und Lob)

Follow-up aus letzter Übungsstunde

- Applausreform
- Gute Nachrichten (und Lob)
- `const` und Funktionen
- Towers of Hanoi

1. Follow-up

1.1. const und Funktionen

const und Funktionen

const und Funktionen

```
1. | void Funktion(const Type& n){           // const als Argument  
    |     // n kann nicht verändert werden  
    | } // sehr oft sinnvoll; muss man kennen!
```

const und Funktionen

1. `void Funktion(const Type& n){ // const als Argument
 // n kann nicht verändert werden
} // sehr oft sinnvoll; muss man kennen!`

2. `const Type& Funktion(const Type& n){ // const als Return
 // n kann nicht verändert werden
 return n; // ursprüngliches n wird returned
} // manchmal sinnvoll; ziemlich spezifisch`

const und Funktionen

1.

```
void Funktion(const Type& n){           // const als Argument
    // n kann nicht verändert werden
} // sehr oft sinnvoll; muss man kennen!
```

2.

```
const Type& Funktion(const Type& n){    // const als Return
    // n kann nicht verändert werden
    return n; // ursprüngliches n wird returned
} // manchmal sinnvoll; ziemlich spezifisch
```

3.

```
void Funktion(Type n) const {           // const-Funktionen
    // n kann verändert werden
    // nichts in der Klasse, der die Funktion angehört,
    // kann verändert werden. [Klassen wurden noch nicht behandelt]
} // oft sinnvoll; wird vielleicht besprochen im Thema "Klassen"
```

struct/classes

1. Follow-up

1.2. Aufgabe "Towers of Hanoi"

Experiment: Die Türme von Hanoi



Links

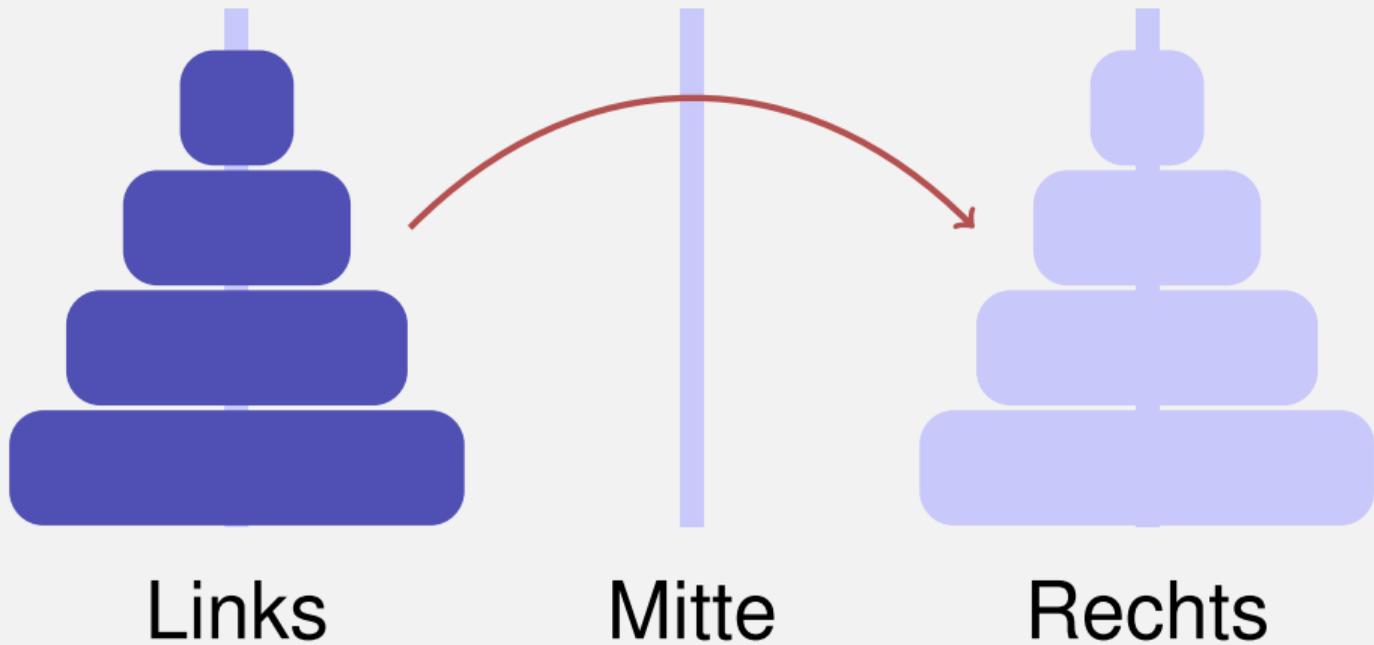


Mitte



Rechts

Experiment: Die Türme von Hanoi



Die Türme von Hanoi - So gehts!



Links



Mitte



Rechts

Die Türme von Hanoi - So gehts!



Links



Mitte



Rechts

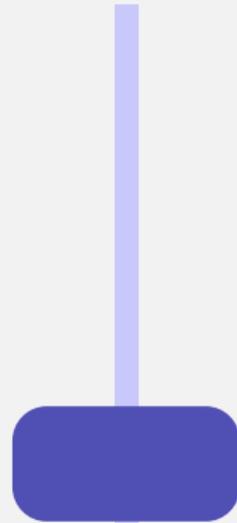
Die Türme von Hanoi - So gehts!



Links



Mitte



Rechts

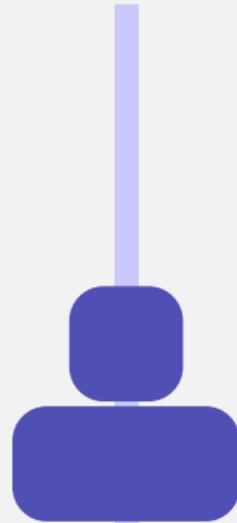
Die Türme von Hanoi - So gehts!



Links

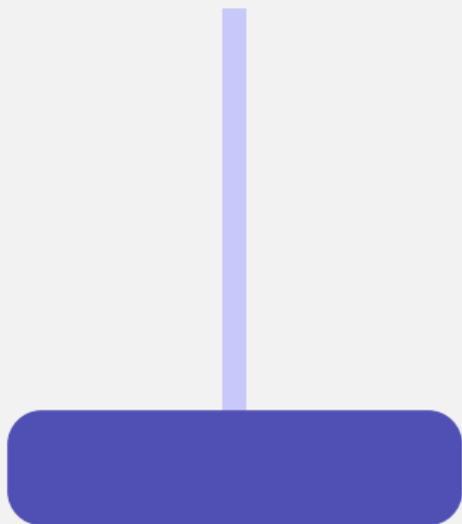


Mitte

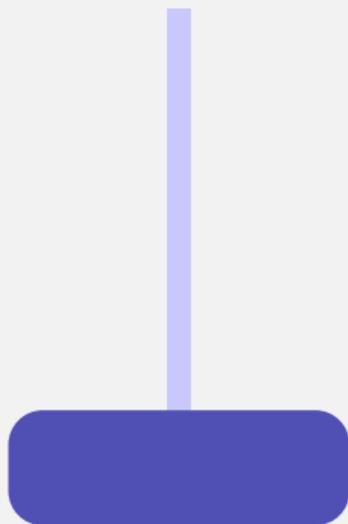


Rechts

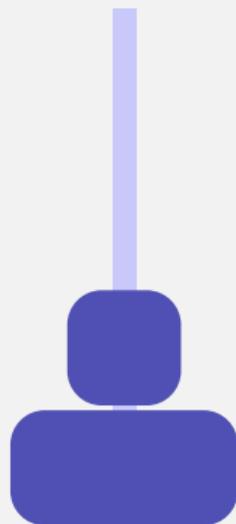
Die Türme von Hanoi - So gehts!



Links

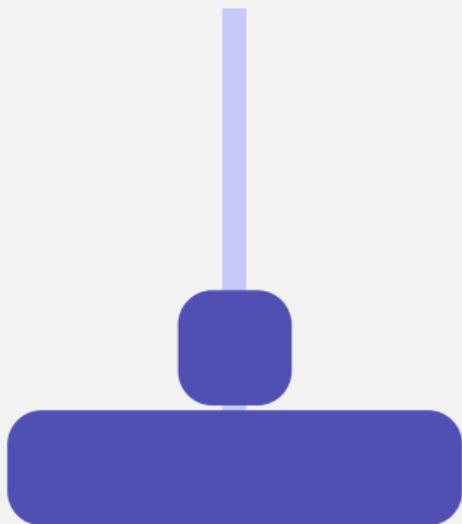


Mitte

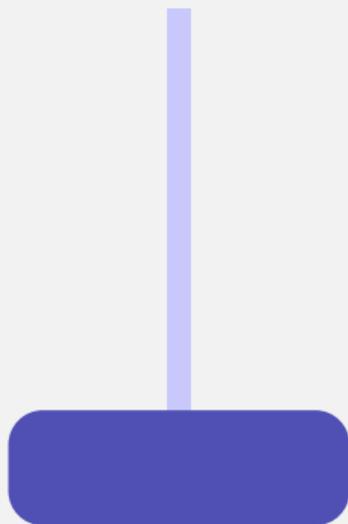


Rechts

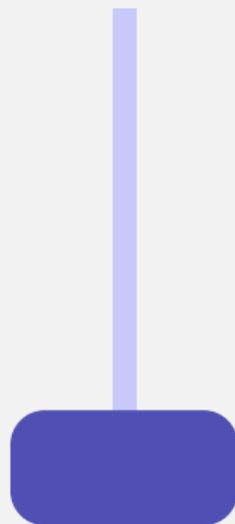
Die Türme von Hanoi - So gehts!



Links

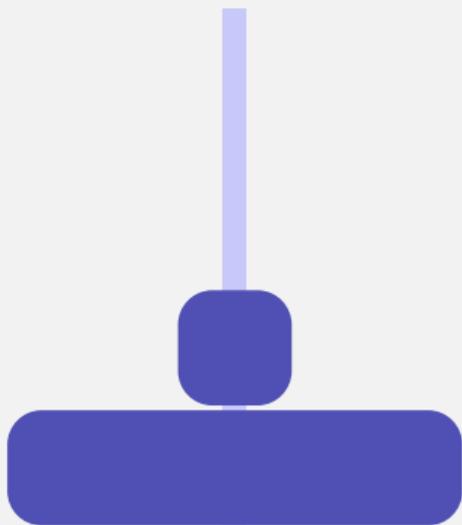


Mitte

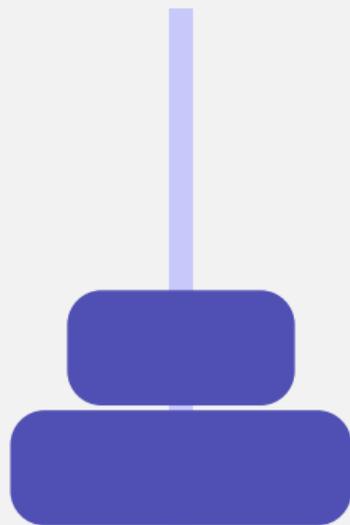


Rechts

Die Türme von Hanoi - So gehts!



Links

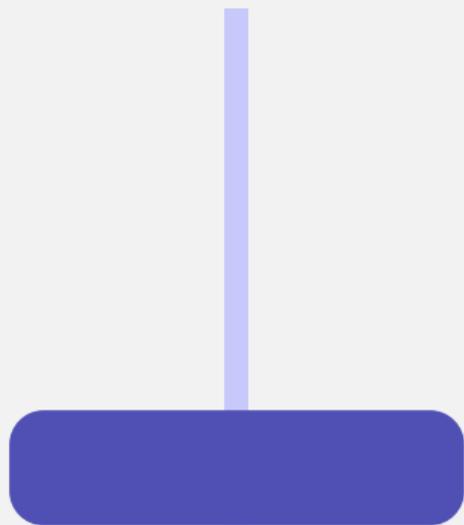


Mitte



Rechts

Die Türme von Hanoi - So gehts!



Links



Mitte

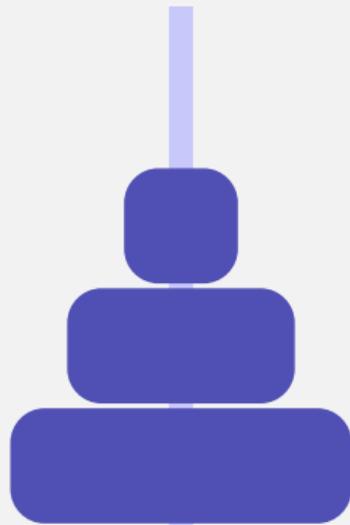


Rechts

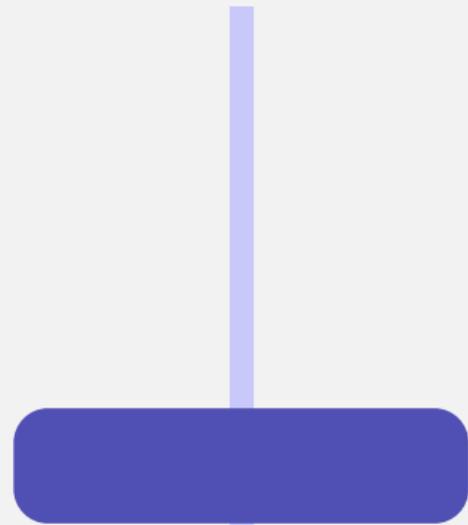
Die Türme von Hanoi - So gehts!



Links



Mitte



Rechts

Die Türme von Hanoi - So gehts!



Links

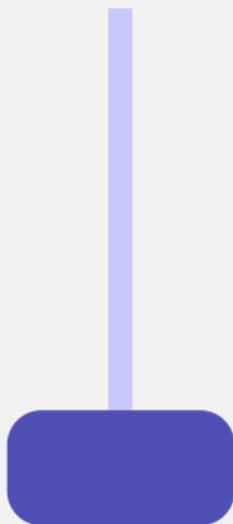


Mitte

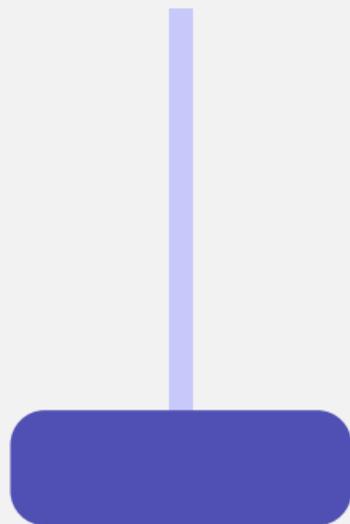


Rechts

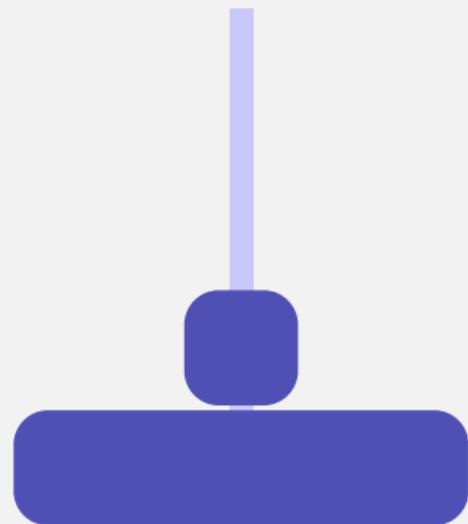
Die Türme von Hanoi - So gehts!



Links

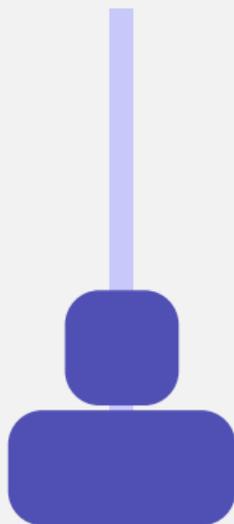


Mitte

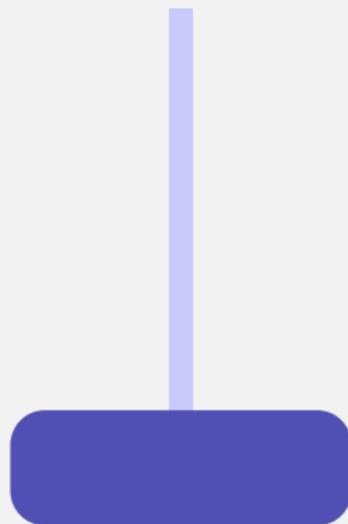


Rechts

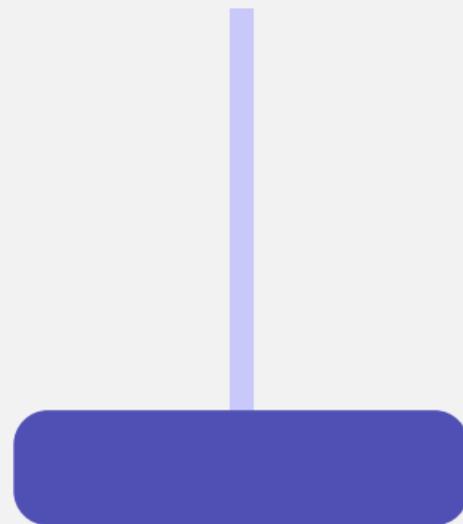
Die Türme von Hanoi - So gehts!



Links

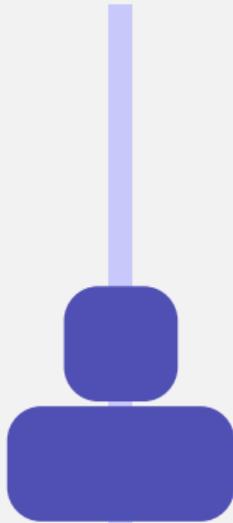


Mitte



Rechts

Die Türme von Hanoi - So gehts!



Links

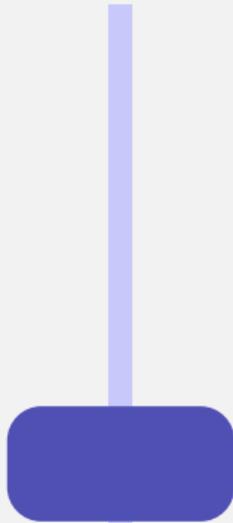


Mitte



Rechts

Die Türme von Hanoi - So gehts!



Links



Mitte



Rechts

Die Türme von Hanoi - So gehts!



Links



Mitte



Rechts

Die Türme von Hanoi - So gehts!



Links



Mitte



Rechts

Die Türme von Hanoi - Rekursiver Lösungsansatz



Links

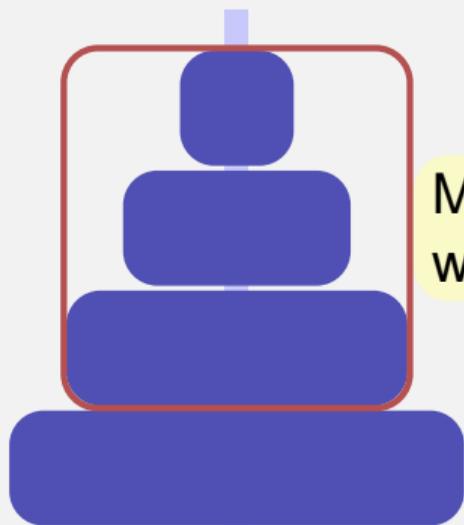


Mitte



Rechts

Die Türme von Hanoi - Rekursiver Lösungsansatz



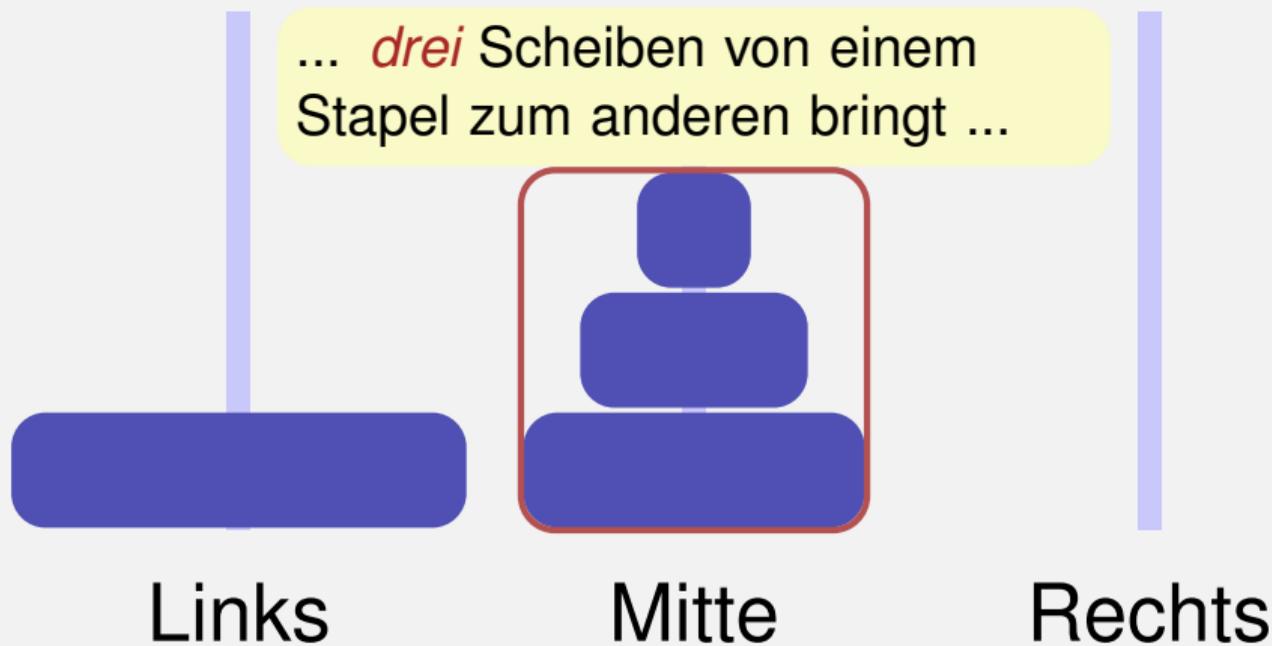
Mal *angenommen*, wir wüssten wie man ...

Links

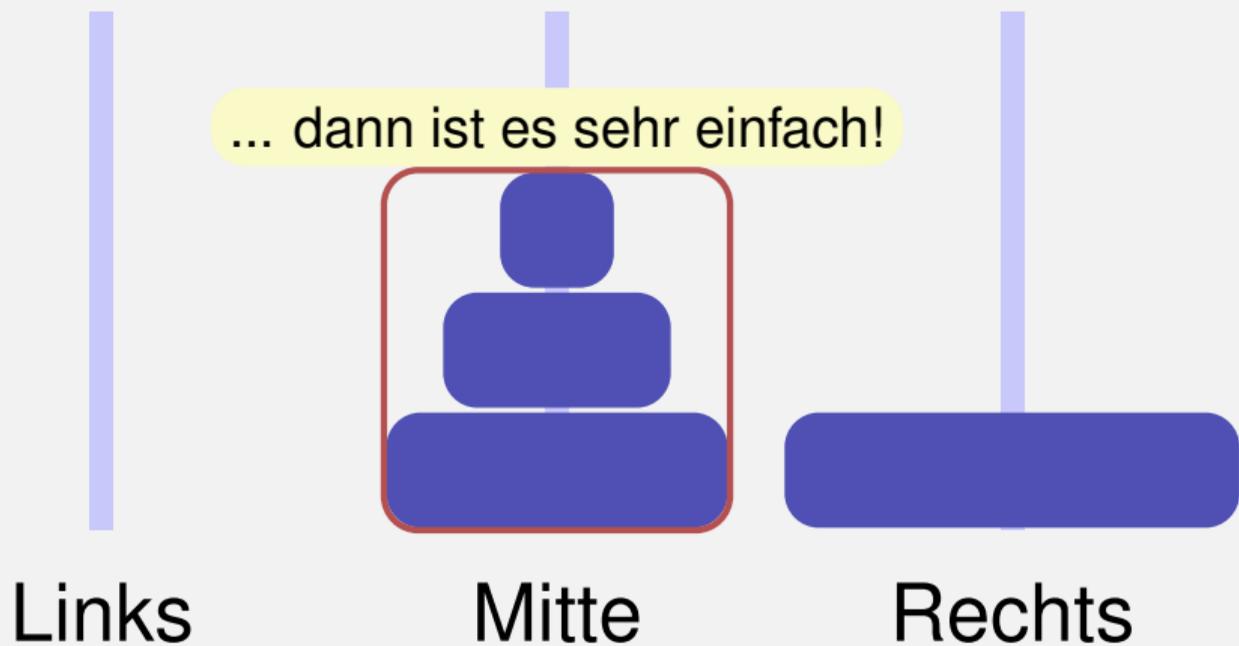
Mitte

Rechts

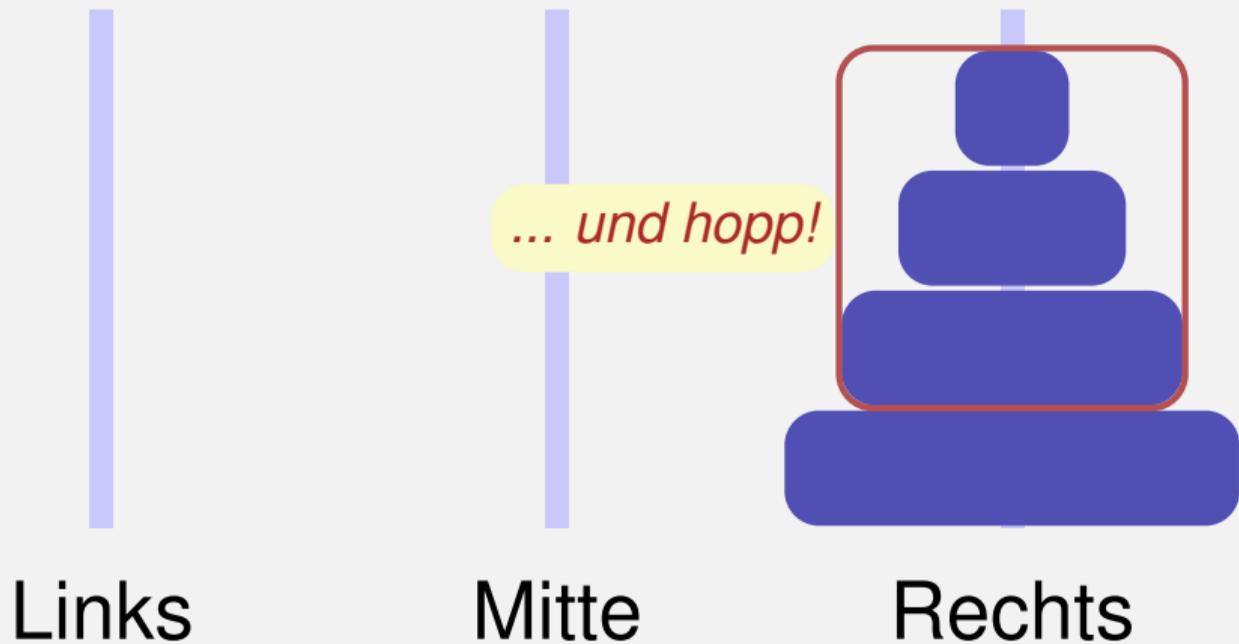
Die Türme von Hanoi - Rekursiver Lösungsansatz



Die Türme von Hanoi - Rekursiver Lösungsansatz



Die Türme von Hanoi - Rekursiver Lösungsansatz



Die Türme von Hanoi - Rekursiver Lösungsansatz



Links

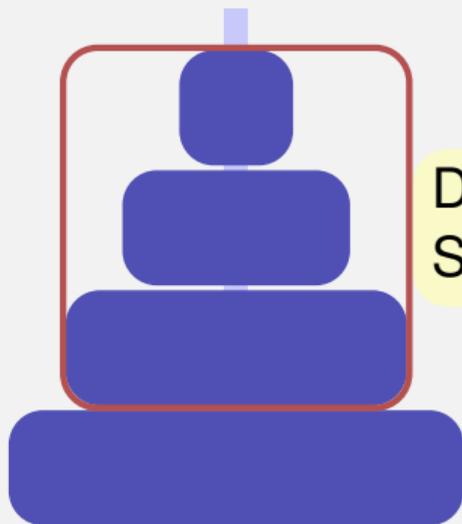


Mitte



Rechts

Die Türme von Hanoi - Rekursiver Lösungsansatz



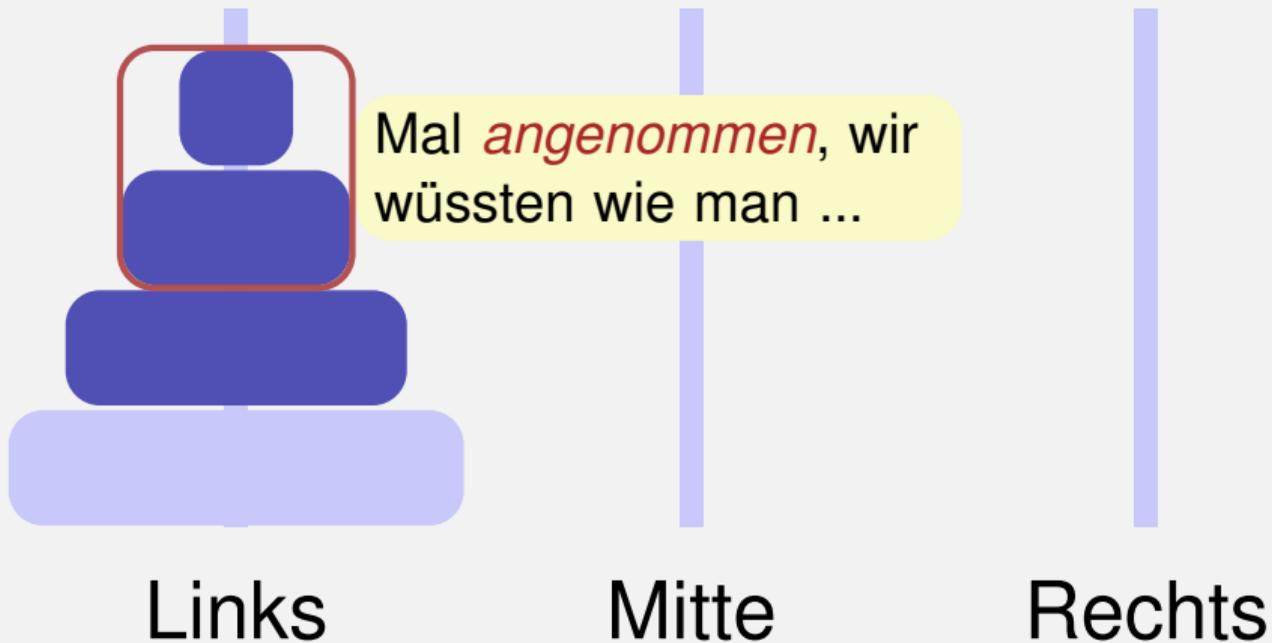
Doch *wie* können wir drei
Scheiben bewegen?

Links

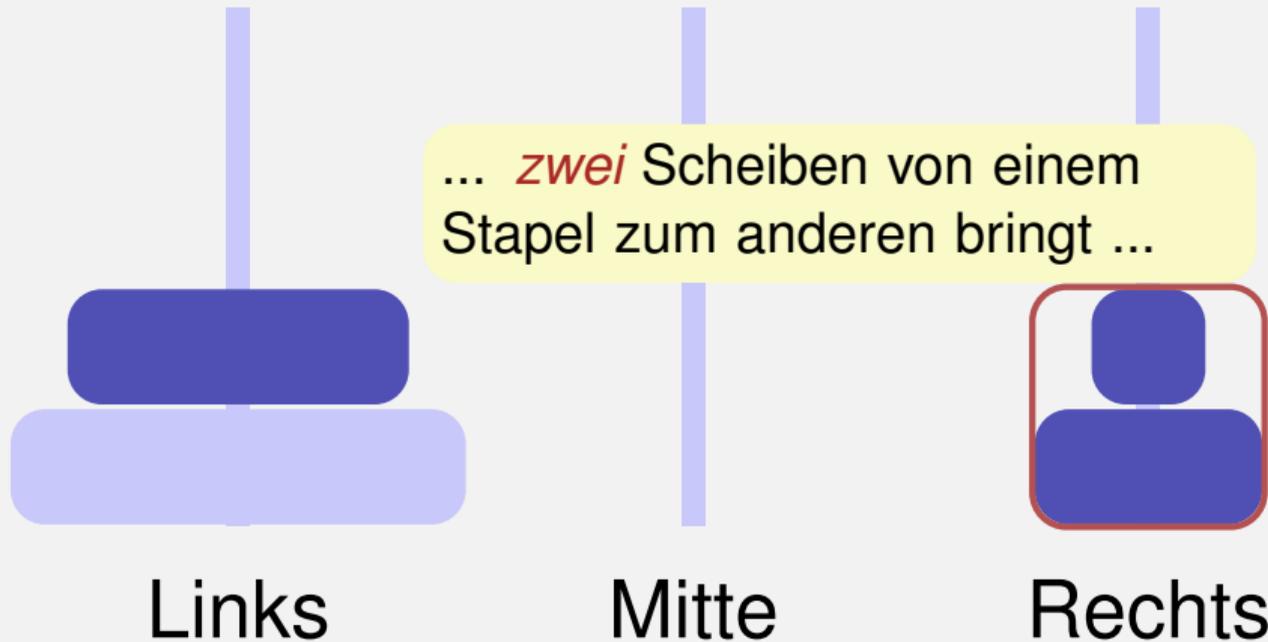
Mitte

Rechts

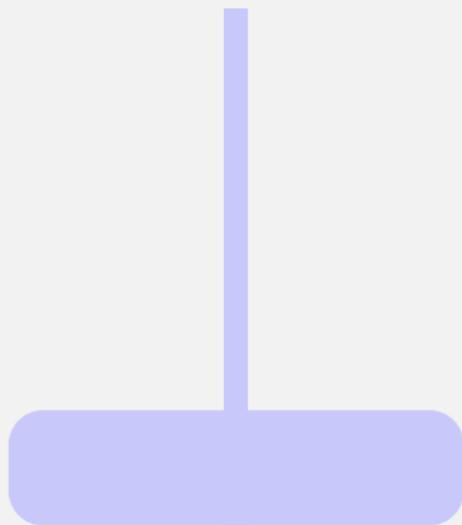
Die Türme von Hanoi - Rekursiver Lösungsansatz



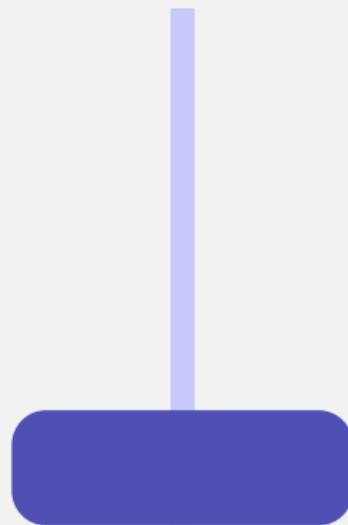
Die Türme von Hanoi - Rekursiver Lösungsansatz



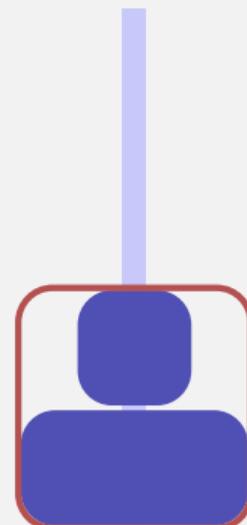
Die Türme von Hanoi - Rekursiver Lösungsansatz



Links

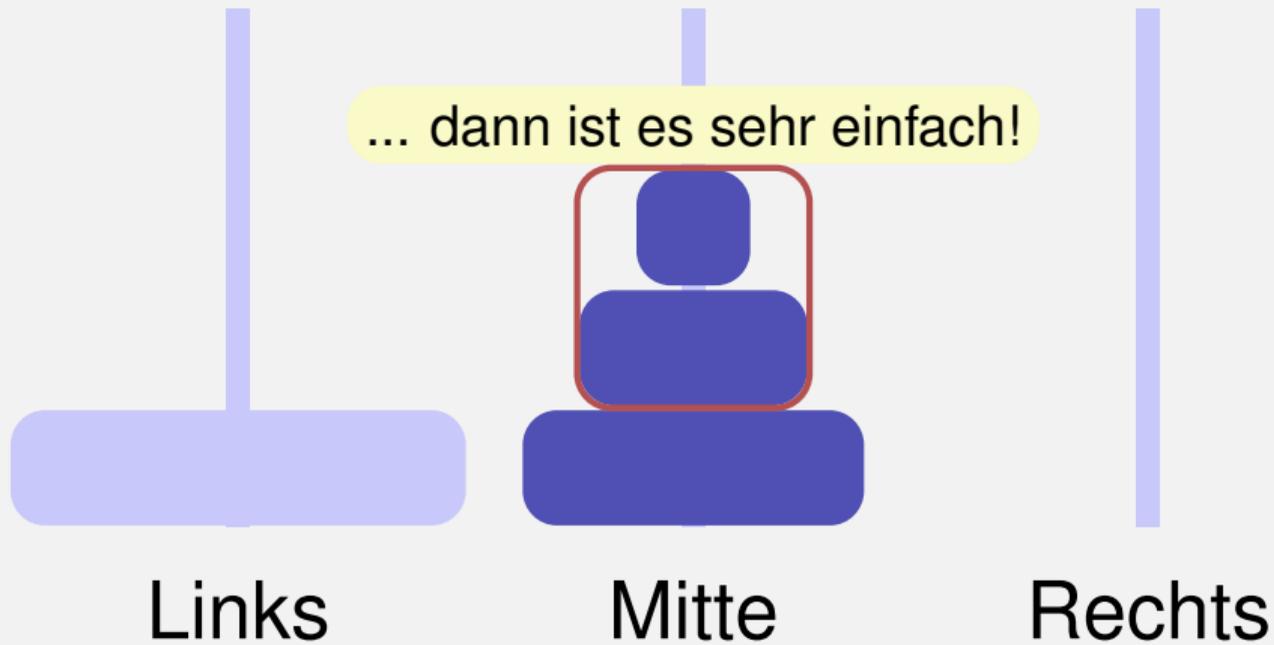


Mitte

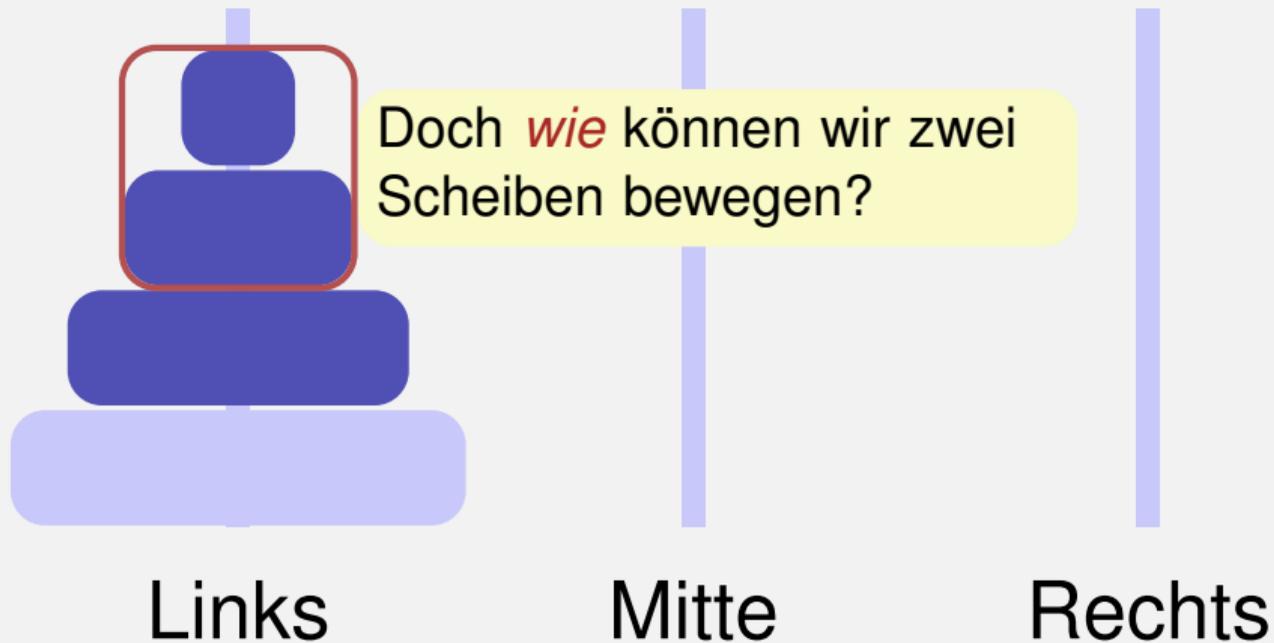


Rechts

Die Türme von Hanoi - Rekursiver Lösungsansatz



Die Türme von Hanoi - Rekursiver Lösungsansatz



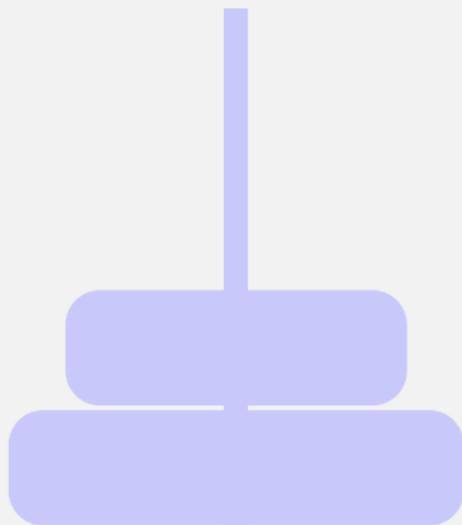
Die Türme von Hanoi - Rekursiver Lösungsansatz



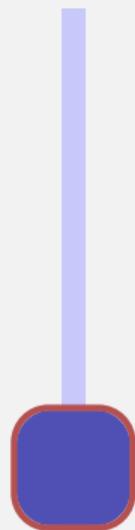
Die Türme von Hanoi - Rekursiver Lösungsansatz



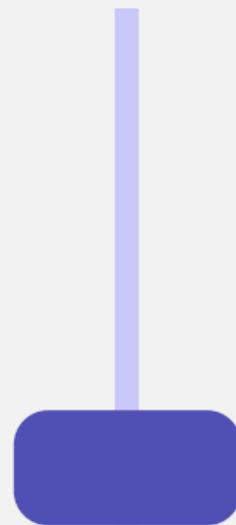
Die Türme von Hanoi - Rekursiver Lösungsansatz



Links



Mitte



Rechts

Die Türme von Hanoi - Rekursiver Lösungsansatz



Aufgabe Towers of Hanoi

- Öffnet "Towers of Hanoi" auf **code expert**

Aufgabe Towers of Hanoi

- Öffnet "Towers of Hanoi" auf **code expert**
- Programmiert eine Lösung

Die Türme von Hanoi - Code



left

middle

right

Bewege 4 Scheiben von left nach right mit Hilfsstapel middle:

```
move(4, "left", "middle", "right")
```

Die Türme von Hanoi - Code

`move(n, src, aux, dst)`

\Rightarrow



- 1 Bewege die obersten $n - 1$ Scheiben von src nach aux mit Hilfsstapel dst :

`move(n - 1, src, dst, aux);`

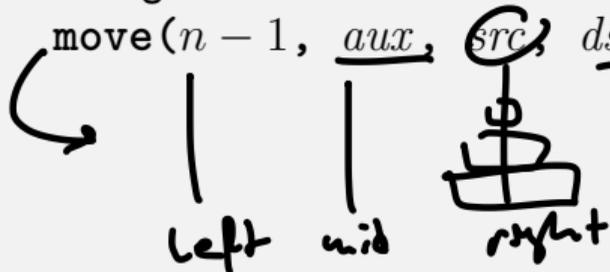
- 2 Bewege 1 Scheibe von src nach dst

`move(1, src, aux, dst);`



- 3 Bewege die obersten $n - 1$ Scheiben von aux nach dst mit Hilfsstapel src :

`move(n - 1, aux, src, dst);`



Die Türme von Hanoi - Code

```
void move(int n, const string &src, const string &aux, const string &dst){  
    if (n == 1) {  
        // base case ('move' the disc)  
        std::cout << src << " --> " << dst << std::endl;  
    } else {  
        // recursive case  
  
    }  
}
```

Die Türme von Hanoi - Code

```
void move(int n, const string &src, const string &aux, const string &dst){  
    if (n == 1) {  
        // base case ('move' the disc)  
        std::cout << src << " --> " << dst << std::endl;  
    } else {  
        // recursive case  
        move(n-1, src, dst, aux);  
  
    }  
}
```

Die Türme von Hanoi - Code

```
void move(int n, const string &src, const string &aux, const string &dst){  
    if (n == 1) {  
        // base case ('move' the disc)  
        std::cout << src << " --> " << dst << std::endl;  
    } else {  
        // recursive case  
        move(n-1, src, dst, aux);  
        move(1, src, aux, dst);  
    }  
}
```

Die Türme von Hanoi - Code

□ liegt in codexprint
(autogeneriert)

```
void move(int n, const string &src, const string &aux, const string &dst){  
    if (n == 1) {  
        // base case ('move' the disc)  
        std::cout << src << " --> " << dst << std::endl;  
    } else {  
        // recursive case  
        move(n-1, src, dst, aux);  
        move(1, src, aux, dst);  
        move(n-1, aux, src, dst);  
    }  
}
```

← passes: $1 \leftarrow n-1$

Die Türme von Hanoi - Code

```
void move(int n, const string &src, const string &aux, const string &dst){
    if (n == 1) {
        // base case ('move' the disc)
        std::cout << src << " --> " << dst << std::endl;
    } else {
        // recursive case
        move(n-1, src, dst, aux);
        move(1, src, aux, dst);
        move(n-1, aux, src, dst);
    }
}

int main() {
    move(4, "left", "middle", "right");
    return 0;
}
```

Die Türme von Hanoi - Code Alternative

```
void move(int n, const string &src, const string &aux, const string &dst){  
    // base case  
    if (n == 0) return;  
  
    // recursive case  
    move(n-1, src, dst, aux);  
    std::cout << src << " --> " << dst << "\n";  
    move(n-1, aux, src, dst);  
}
```

```
int main() {  
    move(4, "left ", "middle", "right ");  
    return 0;  
}
```

Fragen/Unklarheiten?

2. Feedback zu **code** expert

Allgemeines bezüglich **code expert**

Ein paar Vereinfachungen für euren Code¹

```
if(condition == true){  
    // ...  
}
```

→

```
if(condition){  
    //...  
}
```

```
if(condition){  
    return true;  
} else {  
    return false;  
}
```

→

```
return condition;
```

in Funktionen, die einen `bool` zurückgeben

¹Nicht vergessen: Vereinfachungen sind nicht immer besser für die Verständlichkeit

Allgemeines bezüglich **code expert**

Allgemeines bezüglich **code expert**

```
if (condition) {  
    // thing  
}  
else {  
    // other thing  
} // PFUUUUUIII
```

Allgemeines bezüglich **code expert**

```
if (condition) {  
    // thing  
}  
else {                // PFUUUUIII  
    // other thing  
}
```

Setzt die Klammern nach **else** richtig. Wir programmieren hier kein Python!

```
if (condition) {  
    // thing  
} else {  
    // other thing  
}
```

Aufgabe "Pre- and post-conditions"

Aufgabe "Pre- and post-conditions"

```
// PRE n is an integer and larger than 0  
// POST gives amount of digits of n
```

Aufgabe "Pre- and post-conditions"

```
// PRE n is an integer and larger than 0  
// POST gives amount of digits of n
```

Aligned, präziser, einfacher zu lesen:

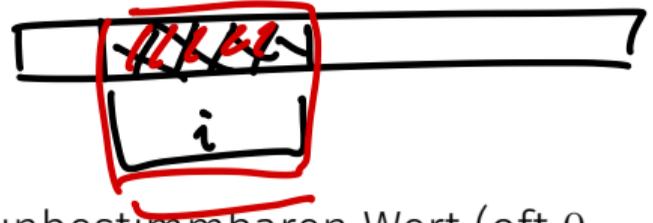
```
// PRE:  n > 0  
// POST: returns number of digits of n (in decmial representation)
```

Aufgabe "Fixing Functions"

Aufgabe "Fixing Functions"

```
int i = 0;
```

```
    i = 5;
```



- Uninitialisierte Variablen haben einen unbestimmbaren Wert (oft 0, aber eben nicht immer!)

```
skl    (i)
```

Aufgabe "Fixing Functions"

- Uninitialisierte Variablen haben einen unbestimmbaren Wert (oft 0, aber eben nicht immer!)
- Benutzt `asserts` um PREs einzuhalten

Wichtige Änderung bezüglich Feedback

Wichtige Änderung bezüglich Feedback

- Aufgrund zu hoher Auslastung, bekommt man ab heute Feedback nur noch auf Anfrage (ausser ich sehe gerade etwas grundlegend falsches)

Wichtige Änderung bezüglich Feedback

- Aufgrund zu hoher Auslastung, bekommt man ab heute Feedback nur noch auf Anfrage (ausser ich sehe gerade etwas grundlegend falsches)
- Diese "Anfrage" kann folgendermassen aussehen und ist gaaaaanz oben im Code anzubringen

```
// FEEDBACK PLEASE  
// - especially regarding lines 12, 13 and 42  
// QUESTIONS  
// - [re: line 42] I was wondering if [...]
```

Wichtige Änderung bezüglich Feedback

- Aufgrund zu hoher Auslastung, bekommt man ab heute Feedback nur noch auf Anfrage (ausser ich sehe gerade etwas grundlegend falsches)
- Diese "Anfrage" kann folgendermassen aussehen und ist gaaaaanz oben im Code anzubringen

```
// FEEDBACK PLEASE  
// - especially regarding lines 12, 13 and 42  
// QUESTIONS  
// - [re: line 42] I was wondering if [...]
```

- TA-Punkte werden noch immer verteilt

Wichtige Änderung bezüglich Feedback

- Aufgrund zu hoher Auslastung, bekommt man ab heute Feedback nur noch auf Anfrage (ausser ich sehe gerade etwas grundlegend falsches)
- Diese "Anfrage" kann folgendermassen aussehen und ist gaaaaanz oben im Code anzubringen

```
// FEEDBACK PLEASE  
// - especially regarding lines 12, 13 and 42  
// QUESTIONS  
// - [re: line 42] I was wondering if [...]
```

- TA-Punkte werden noch immer verteilt
- Falls irgendwo die XP auf 0 gesetzt werden müssen, wird im Feedback erwähnt sein weshalb

Fragen bezüglich **code expert** eurerseits?

3. Lernziele

Ziele für Heute

Lernziele

Ziele für Heute

Lernziele

- Code, der **structs** enthält schreiben und verstehen können
- Code, der Rekursion enthält schreiben und verstehen können

Ziele für Heute

Lernziele

- Code, der **structs** enthält schreiben und verstehen können
- Code, der Rekursion enthält schreiben und verstehen können
- Durch "Towers of Hanoi" leicht traumatisiert werden

4. Zusammenfassung

5. Structs

Beispiel für Structs

Beispiel für Structs

```
struct strange {
    int n;
    bool b;
    std::vector<int> a = std::vector<int>(0);
};

int main () {
    strange x = {1, true, {1,2,3}};
    strange y = x; // all elements are copied
    std::cout << y.n << " " << y.a[2] << "\n"; // outputs: 1 3
    return 0;
}
```

5. Structs

5.1. Aufgabe "Geometry Exercise"

Aufgabe "Geometry Exercise"

- Öffnet "Geometry Exercise" auf **code expert**

Aufgabe "Geometry Exercise"

- Öffnet "Geometry Exercise" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet

Aufgabe "Geometry Exercise"

- Öffnet "Geometry Exercise" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet
- Zeit für Group Programming!

6. Rekursion

6. Rekursion

6.1. Aufgabe "Power Set"

Aufgabe "Power Set"

Rekapitulation

- Ein Potenzmenge ist die Menge aller Teilmengen

Aufgabe "Power Set"

Rekapitulation

- Ein Potenzmenge ist die Menge aller Teilmengen

$$\mathcal{P}(S) := \{X \mid X \subseteq S\}$$

Aufgabe "Power Set"

Rekapitulation

- Ein Potenzmenge ist die Menge aller Teilmengen

$$\mathcal{P}(S) := \{X \mid X \subseteq S\}$$

- Beispiel:
 - Gegeben sei die Menge $A = \{a, b, c\}$

Aufgabe "Power Set"

Rekapitulation

- Ein Potenzmenge ist die Menge aller Teilmengen

$$\mathcal{P}(S) := \{X \mid X \subseteq S\}$$

$$|\mathcal{P}(S)| = 2^{|S|}$$

- Beispiel:

- Gegeben sei die Menge $A = \{a, b, c\}$
- Die Potenzmenge ist

$$\mathcal{P}(A) = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$$

Einführung zu set.h

- set ist eine selbstgebastelter Typ! (eine **class**)
- Wie funktioniert er? Seht selbst in set.h!

```
template <typename T>
class Set {
    public:
        Set(const Set& other);
        // Creates an empty set
        Set();
        // Creates a new set from a set of elements
        Set(const std::set<T>& elements);
        // Creates a new set from a single element
        Set(T element);
        // ...
};
```

Aufgabe Power Set

- Öffnet "Power Set" auf **code expert**

Aufgabe Power Set

- Öffnet "Power Set" auf **code expert**
- Überlegt euch, wie ihr das konzeptionell lösen könnt

Aufgabe Power Set

- Öffnet "Power Set" auf **code expert**
- Überlegt euch, wie ihr das konzeptionell lösen könnt
- Programmiert eine Lösung

Aufgabe Power Set

- Öffnet "Power Set" auf **code expert**
- Überlegt euch, wie ihr das konzeptionell lösen könnt
- Programmiert eine Lösung
- Tipp: Die Funktionalitäten vom Typ `set` findet ihr im `main.cpp`-file
- Mögliche Leitfragen: Für welche (einfachen) Fälle kennen wir die Lösung immer? Gibt es ein Muster, dem die Potenzmengen folgen, wenn ein weiteres Element dazukommt?

~ wie füge ich ein Element hinzu?
"Wie initialisiere ich eine Menge mit nur einem Element", etc...

Lösung zu "Power Set" (Konzeptionell)

Gegeben: $\{a, b, c, d\}$

²Hier kommt der Vertrauensvorschuss der Rekursion (*Recursive Leap of Faith*) zum Tragen

Lösung zu "Power Set" (Konzeptionell)

Gegeben: $\{a, b, c, d\}$

// set has at least 1 element -> split set into two sets

²Hier kommt der Vertrauensvorschuss der Rekursion (*Recursive Leap of Faith*) zum Tragen

Lösung zu "Power Set" (Konzeptionell)

Gegeben: $\{a, b, c, d\}$

// set has at least 1 element -> split set into two sets

$\{a\}, \quad \{b, c, d\}$

²Hier kommt der Vertrauensvorschuss der Rekursion (*Recursive Leap of Faith*) zum Tragen

Lösung zu "Power Set" (Konzeptionell)

Gegeben: $\{a, b, c, d\}$

// set has at least 1 element -> split set into two sets

$\{a\}, \quad \{b, c, d\}$

// get power set for remaining subset²

²Hier kommt der Vertrauensvorschuss der Rekursion (*Recursive Leap of Faith*) zum Tragen

Lösung zu "Power Set" (Konzeptionell)

Gegeben: $\{a, b, c, d\}$

// set has at least 1 element -> split set into two sets

$$\{a\}, \quad \{b, c, d\}$$

// get power set for remaining subset²

$$\mathcal{P}(\{b, c, d\}) = \{\{\}, \{b\}, \{c\}, \{d\}, \{b, c\}, \dots\}$$

²Hier kommt der Vertrauensvorschuss der Rekursion (*Recursive Leap of Faith*) zum Tragen

Lösung zu "Power Set" (Konzeptionell)

Gegeben: $\{a, b, c, d\}$

// set has at least 1 element -> split set into two sets

$$\{a\}, \quad \{b, c, d\}$$

// get power set for remaining subset²

$$\mathcal{P}(\{b, c, d\}) = \{\{\}, \{b\}, \{c\}, \{d\}, \{b, c\}, \dots\}$$

// init result with power set of remaining subset

²Hier kommt der Vertrauensvorschuss der Rekursion (*Recursive Leap of Faith*) zum Tragen

Lösung zu "Power Set" (Konzeptionell)

Gegeben: $\{a, b, c, d\}$

// set has at least 1 element -> split set into two sets

$$\{a\}, \quad \{b, c, d\}$$

// get power set for remaining subset²

$$\mathcal{P}(\{b, c, d\}) = \{\{\}, \{b\}, \{c\}, \{d\}, \{b, c\}, \dots\}$$

// init result with power set of remaining subset

$$\text{result} \leftarrow \{\{\}, \{b\}, \{c\}, \{d\}, \{b, c\}, \dots\}$$

²Hier kommt der Vertrauensvorschuss der Rekursion (*Recursive Leap of Faith*) zum Tragen

Lösung zu "Power Set" (Konzeptionell)

Gegeben: $\{a, b, c, d\}$

// set has at least 1 element -> split set into two sets

$$\{a\}, \quad \{b, c, d\}$$

// get power set for remaining subset²

$$\mathcal{P}(\{b, c, d\}) = \{\{\}, \{b\}, \{c\}, \{d\}, \{b, c\}, \dots\}$$

// init result with power set of remaining subset

$$\text{result} \leftarrow \{\{\}, \{b\}, \{c\}, \{d\}, \{b, c\}, \dots\}$$

// add first element to every set in the powerset

²Hier kommt der Vertrauensvorschuss der Rekursion (*Recursive Leap of Faith*) zum Tragen

Lösung zu "Power Set" (Konzeptionell)

Gegeben: $\{a, b, c, d\}$

// set has at least 1 element -> split set into two sets

$$\{a\}, \quad \{b, c, d\}$$

// get power set for remaining subset²

$$\mathcal{P}(\{b, c, d\}) = \{\{\}, \{b\}, \{c\}, \{d\}, \{b, c\}, \dots\}$$

// init result with power set of remaining subset

$$\text{result} \leftarrow \{\{\}, \{b\}, \{c\}, \{d\}, \{b, c\}, \dots\}$$

// add first element to every set in the powerset

$$\left\{ \begin{array}{l} \{\}, \{b\}, \{c\}, \{d\}, \{b, c\}, \dots, \\ \{a\}, \{a, b\}, \{a, c\}, \{a, d\}, \{a, b, c\}, \dots \end{array} \right\}$$

²Hier kommt der Vertrauensvorschuss der Rekursion (*Recursive Leap of Faith*) zum Tragen

Lösung zu "Power Set" (Basisfall)

Lösung zu "Power Set" (Basisfall)

```
SetOfCharSets power_set(const CharSet& set) {  
    // base case: empty set  
    if (set.size() == 0) {  
        return SetOfCharSets(CharSet());  
    }  
}
```

Lösung zu "Power Set"

```
// set has at least 1 element -> split set into two sets.
CharSet first_element_subset = CharSet(set.at(0));
CharSet remaining_subset = set - first_element_subset;

// get power set for remaining subset
SetOfCharSets remaining_subset_power_set = power_set(remaining_subset);

// init result with power set of remaining subset
SetOfCharSets result = remaining_subset_power_set;

// add first element to every set in the powerset
for (unsigned int i = 0; i < remaining_subset_power_set.size(); ++i) {
    result.insert(first_element_subset + remaining_subset_power_set.at(i));
}

return result;
```

Fragen/Unklarheiten?

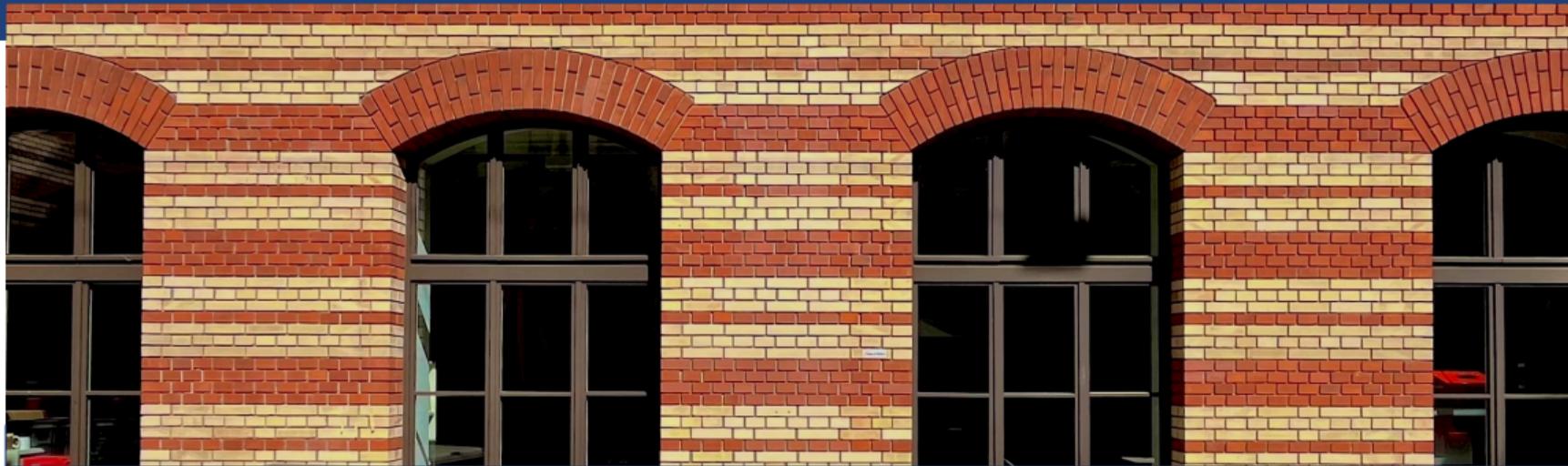
7. Outro

Allgemeine Fragen?

Bis zum nächsten Mal

□ @Dat:
script/lec notes
schneller online pls

Schöne Woche noch!



Übungsstunde — Informatik — 09

Adel Gavranović

Structs, Classes, Operator-overloading, Iteratoren

Übersicht

Follow-up
Klassen und Operator-Überladung
Aufgabe "Tribool"
Iteratoren
Aufgabe "Find Max"
Rekursion



n.ethz.ch/~agavranovic

[Material](#)

[Webpage](#)

[Mail](#)

1. Follow-up

Follow-up aus letzter Übungsstunde

Follow-up aus letzter Übungsstunde

- (ES08::S7 ff.) Im "die Türme von Hanoi"-Codebeispiel gab es tatsächlich einen Bug beim Autograder falls man den Code falsch implementiert, i.e.

```
if(n == 1){ // BASE CASE
    std::cout << src << " --> " << dst << std::endl;
    return;
} else { // RECURSIVE CASE
    move(n-1, src, dst, aux);
    move(n-1, src, aux, dst); // <-- difference from MS:
    move(n-1, aux, src, dst); // "n-1" not just "1"
}
```

ergibt eine falsche Lösung für $n = 3$, aber der Autograder markiert sie als richtig

2. Feedback zu **code** expert

Fragen bezüglich **code expert** eurerseits?

3. Lernziele

Ziele

- eigene Klassen definieren können
- Operatoren für bereits definierte Klassen überladen können
- Iteratoren verwenden können

4. Zusammenfassung

5. Klassen und Operator-Überladung

Funktionen voneinander differenzieren

Funktionen voneinander differenzieren

Es ist möglich, dass zwei Funktionen den gleichen Namen haben, solange der Compiler eine andere Möglichkeit hat, sie zu unterscheiden. Die einzigen möglichen Kriterien Funktionen zu unterscheiden sind also:

Funktionen voneinander differenzieren

Es ist möglich, dass zwei Funktionen den gleichen Namen haben, solange der Compiler eine andere Möglichkeit hat, sie zu unterscheiden. Die einzigen möglichen Kriterien Funktionen zu unterscheiden sind also:

- Namen der Funktionen
- Anzahl der Funktionsargumente
- Typen der Funktionsargumente

Putting the *Fun* in *Function* I

Wird dies zu einem **Compilerfehler** führen?

```
int fun1(const int a){  
    // ...  
}  
  
int fun1(const int a, const int b){  
    // ...  
}
```

Putting the *Fun* in *Function* I

Wird dies zu einem **Compilerfehler** führen?

```
int fun1(const int a){  
    // ...  
}  
  
int fun1(const int a, const int b){  
    // ...  
}
```

Antwort: Nein, weil

Putting the *Fun* in *Function* I

Wird dies zu einem **Compilerfehler** führen?

```
int fun1(const int a){  
    // ...  
}  
  
int fun1(const int a, const int b){  
    // ...  
}
```

Antwort: Nein, weil die beiden Funktionen unterschiedlich viele Argumente besitzen (1 vs 2)

Putting the *Fun* in *Function* II

Wird dies zu einem **Compilerfehler** führen?

```
int fun2(const int a){  
    // ...  
}  
  
int fun2(const float a){  
    // ...  
}
```

Putting the *Fun* in *Function* II

Wird dies zu einem **Compilerfehler** führen?

```
int fun2(const int a){  
    // ...  
}  
  
int fun2(const float a){  
    // ...  
}
```

Antwort: Nein, weil

Putting the *Fun* in *Function* II

Wird dies zu einem **Compilerfehler** führen?

```
int fun2(const int a){  
    // ...  
}  
  
int fun2(const float a){  
    // ...  
}
```

Antwort: Nein, weil die beiden Funktionen unterschiedliche Argumenttypen besitzen (`int` vs `float`)

Putting the *Fun* in *Function* III

Wird dies zu einem **Compilerfehler** führen?

```
int fun3(const int a){  
    // ...  
}  
  
int fun3(const int b){  
    // ...  
}
```

Putting the *Fun* in *Function* III

Wird dies zu einem **Compilerfehler** führen?

```
int fun3(const int a){  
    // ...  
}  
  
int fun3(const int b){  
    // ...  
}
```

Antwort: Ja, weil

Putting the *Fun* in *Function* III

Wird dies zu einem **Compilerfehler** führen?

```
int fun3(const int a){  
    // ...  
}  
  
int fun3(const int b){  
    // ...  
}
```

Antwort: Ja, weil die beiden Funktionen sich nicht in der Anzahl oder Typ(en) ihrer Argumente unterscheiden.

Putting the *Fun* in *Function* III

Wird dies zu einem **Compilerfehler** führen?

```
int fun3(const int a){  
    // ...  
}  
  
int fun3(const int b){  
    // ...  
}
```

Antwort: Ja, weil die beiden Funktionen sich nicht in der Anzahl oder Typ(en) ihrer Argumente unterscheiden.

Merke: Die Namen der Funktionsparameter sind für den Compiler irrelevant!

Putting the *Fun* in *Function* IV

Wird dies zu einem **Compilerfehler** führen?

```
int fun4(const int a){  
    // ...  
}  
  
double fun4(const int a){  
    // ...  
}
```

Putting the *Fun* in *Function* IV

Wird dies zu einem **Compilerfehler** führen?

```
int fun4(const int a){  
    // ...  
}  
  
double fun4(const int a){  
    // ...  
}
```

Antwort: Ja, weil

Putting the *Fun* in *Function* IV

Wird dies zu einem **Compilerfehler** führen?

```
int fun4(const int a){  
    // ...  
}  
  
double fun4(const int a){  
    // ...  
}
```

Antwort: Ja, weil die beiden Funktionen sich nicht in der Anzahl oder Typ(en) ihrer Argumente unterscheiden.

Putting the *Fun* in *Function* IV

Wird dies zu einem **Compilerfehler** führen?

```
int fun4(const int a){
    // ...
}

double fun4(const int a){
    // ...
}
```

Antwort: Ja, weil die beiden Funktionen sich nicht in der Anzahl oder Typ(en) ihrer Argumente unterscheiden.

Merke: Die Wiedergabetypen sind für den Compiler irrelevant!

Putting the *Fun* in *Function V*

Wird dies zu einem **Compilerfehler** führen?

```
int fun5(const int a){  
    // ...  
}  
  
int fun6(const int a){  
    // ...  
}
```

Putting the *Fun* in *Function V*

Wird dies zu einem **Compilerfehler** führen?

```
int fun5(const int a){  
    // ...  
}  
  
int fun6(const int a){  
    // ...  
}
```

Antwort: Nein, weil

Putting the *Fun* in *Function V*

Wird dies zu einem **Compilerfehler** führen?

```
int fun5(const int a){  
    // ...  
}  
  
int fun6(const int a){  
    // ...  
}
```

Antwort: Nein, weil die beiden Funktionen unterschiedlich Namen tragen

Genau mein Typ

```
void out(const int i){
    std::cout << i << " (int)\n";
}
void out(const double i){
    std::cout << i << " (double)\n";
}

int main(){
    out(3.5);
    out(2);
    out(2.0);
    out(0);
    out(0.0);
    return 0;
}
```

Wie wird die Ausgabe aussehen?

Genau mein Typ

```
void out(const int i){
    std::cout << i << " (int)\n";
}
void out(const double i){
    std::cout << i << " (double)\n";
}

int main(){
    out(3.5);
    out(2);
    out(2.0);
    out(0);
    out(0.0);
    return 0;
}
```

Wie wird die Ausgabe aussehen?

■ 3.5 (double)

Genau mein Typ

```
void out(const int i){
    std::cout << i << " (int)\n";
}
void out(const double i){
    std::cout << i << " (double)\n";
}

int main(){
    out(3.5);
    out(2);
    out(2.0);
    out(0);
    out(0.0);
    return 0;
}
```

Wie wird die Ausgabe aussehen?

- 3.5 (double)
- 2 (int)

Genau mein Typ

```
void out(const int i){
    std::cout << i << " (int)\n";
}
void out(const double i){
    std::cout << i << " (double)\n";
}

int main(){
    out(3.5);
    out(2);
    → out(2.0);
    out(0);
    out(0.0);
    return 0;
}
```

Wie wird die Ausgabe aussehen?

- 3.5 (double)
- 2 (int)
- 2 (double)

↳ weil .0 nicht
auf console
ausgegeben wird

Genau mein Typ

```
void out(const int i){
    std::cout << i << " (int)\n";
}
void out(const double i){
    std::cout << i << " (double)\n";
}

int main(){
    out(3.5);
    out(2);
    out(2.0);
    out(0);
    out(0.0);
    return 0;
}
```

Wie wird die Ausgabe aussehen?

- 3.5 (double)
- 2 (int)
- 2 (double)
- 0 (int)

Genau mein Typ

```
void out(const int i){
    std::cout << i << " (int)\n";
}
void out(const double i){
    std::cout << i << " (double)\n";
}

int main(){
    out(3.5);
    out(2);
    out(2.0);
    out(0);
    out(0.0);
    return 0;
}
```

Wie wird die Ausgabe aussehen?

- 3.5 (double)
- 2 (int)
- 2 (double)
- 0 (int)
- 0 (double)

Fragen/Unklarheiten?

6. Aufgabe "Tribool"

Tribool als Logik-Objekt

NOT(A)		AND(A,B)				OR(A,B)					
A	$\neg A$	$A \wedge B$		B			$A \vee B$		B		
				F	U	T			F	U	T
F	T	A	F	F	F	F	A	F	F	U	T
U	U		U	F	U	U	U	U	U	U	T
T	F		T	F	U	T	T	T	T	T	T

F = FALSE, U = UNKNOWN, T = TRUE

Tribool als Logik-Objekt

Tribool operator && (const Tribool & A, const Tribool & B)

NOT(A)

A	$\neg A$
F	T
U	U
T	F

AND(A,B)

$A \wedge B$		B		
		F	U	T
A	F	F	F	F
	U	F	U	U
	T	F	U	T

OR(A,B)

$A \vee B$		B		
		F	U	T
A	F	F	U	T
	U	U	U	T
	T	T	T	T

F = FALSE, U = UNKNOWN, T = TRUE

- Wie könnten wir das in C++ implementieren?
- Welche Operationen und Werte brauchen wir?

Exercise "Tribool"

```
class Tribool {  
private:  
    // 0 means false, 1 means unknown, 2 means true.  
    unsigned int value; // INV: value in {0, 1, 2}.  
public:  
    // ...  
};
```

Exercise "Tribool"

```
class Tribool {
private:
    // ...
public:
    // Constructor 1 (passing a numerical value)
    // PRE: value in {0, 1, 2}.
    // POST: tribool false if value was 0, unknown if 1, and true if 2.
    Tribool(unsigned int value_int);
    // TODO: add the definition in tribool.cpp

    // Constructor 2 (passing a string value)
    // PRE: value in {"true", "false", "unknown"}.
    // POST: tribool false, true or unknown according to the input.
    // TODO: add declaration here and the definition in tribool.cpp
    // ...
};
```

Exercise "Tribool"

```
class Tribool {
private:
    // ...
public:
    // ...
    // Member function string()
    // POST: Return the value as string
    // TODO: add declaration here and the definition in tribool.cpp

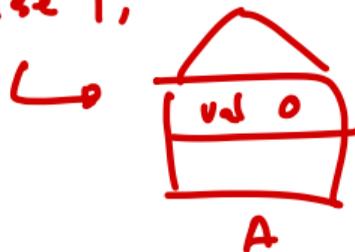
    // Operator && overloading
    // POST: returns this AND other
    // TODO: add declaration here and the definition in tribool.cpp
};
```

Exercise "Tribool"

Wo fangen wir überhaupt an?

1. Erster (`int`) Constructor
2. Zweiter (`std::string`) Constructor
3. Memberfunktion `string()` implementieren
4. Logisches UND als Operatoren implementieren

Tribool A("false");



Exercise "Tribool"

Wo fangen wir überhaupt an?

1. Erster (`int`) Constructor
2. Zweiter (`std::string`) Constructor
3. Memberfunktion `string()` implementieren
4. Logisches UND als Operatoren implementieren

Wohin mit all dem?

- Deklarationen ins `Tribool.h`
- Definitionen ins `Tribool.cpp`
 - Mit Out-of-Class-Definitionen mittels Scope Resolution Operator (`::`)

Let's Code (gemeinsam)!

- Öffnet "Tribool" auf **code expert**

Let's Code (gemeinsam)!

- Öffnet "Tribool" auf **code expert**
- Wir machen eine live Coding-Session

Exercise "Tribool" Konzepte

Folgenden Konzepten und Keywords sind wir beim Lösen dieser Aufgabe begegnet:

Exercise "Tribool" Konzepte

Folgenden Konzepten und Keywords sind wir beim Lösen dieser Aufgabe begegnet:

- Classes und Structs
- Visibility (*private vs. public*)
- Operator Overloading (*operator*)
- Deklaration vs Definition
- Out-of-Class-Definitionen *Tribool::function*
- `const` Funktionen
- Konstruktoren ("C-tors")
- Member Initializer Lists
- ...

Fragen/Unklarheiten?

7. Iteratoren

Was sind Iteratoren überhaupt?

- Iteratoren werden verwendet, um durch Elemente in einem Container zu iterieren

std::ectors

std::set



¹<https://en.cppreference.com/w/cpp/container>

Was sind Iteratoren überhaupt?

- Iteratoren werden verwendet, um durch Elemente in einem Container zu iterieren
- Und was sind Container?
 - Container sind Objekte, die zum Speichern von Sammlungen von Elementen verwendet werden
 - Zu den gängigen C++-Containern gehören:

¹<https://en.cppreference.com/w/cpp/container>

Was sind Iteratoren überhaupt?

- Iteratoren werden verwendet, um durch Elemente in einem Container zu iterieren
- Und was sind Container?
 - Container sind Objekte, die zum Speichern von Sammlungen von Elementen verwendet werden
 - Zu den gängigen C++-Containern gehören:
 - ▶ `std::vector`
 - ▶ `std::set`
 - ▶ `std::list`

¹<https://en.cppreference.com/w/cpp/container>

Was sind Iteratoren überhaupt?

- Iteratoren werden verwendet, um durch Elemente in einem Container zu iterieren
- Und was sind Container?
 - Container sind Objekte, die zum Speichern von Sammlungen von Elementen verwendet werden
 - Zu den gängigen C++-Containern gehören:
 - ▶ `std::vector`
 - ▶ `std::set`
 - ▶ `std::list`
 - Eine vollständige Liste der Container der C++-Standardbibliothek ist hier zu finden,¹ aber die meisten sind für uns nicht relevant

¹<https://en.cppreference.com/w/cpp/container>

Iteratoren auf Containern verwenden

Sehr einfach und absichtlich immer gleich!

Gegeben sei ein Container names C

²Sehr nützlich für unhandliche Wiedergabetypen

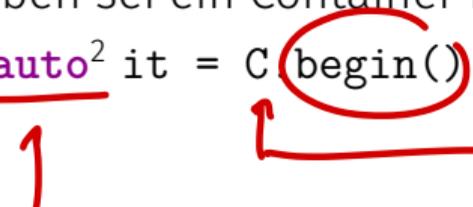
³PTE: Past-the-End

Iteratoren auf Containern verwenden

Sehr einfach und absichtlich immer gleich!

Gegeben sei ein Container names C

■ auto² it = C.begin()



std::vector::iterator

²Sehr nützlich für unhandliche Wiedergabetypen

³PTE: Past-the-End

Iteratoren auf Containern verwenden

Sehr einfach und absichtlich immer gleich!

Gegeben sei ein Container names C

- `auto2 it = C.begin()`
Iterator, der auf das erste Element zeigt
- `auto it = C.end()`

²Sehr nützlich für unhandliche Wiedergabetypen

³PTE: Past-the-End

Iteratoren auf Containern verwenden

Sehr einfach und absichtlich immer gleich!

Gegeben sei ein Container names C

- `auto2 it = C.begin()`

Iterator, der auf das erste Element zeigt

- `auto it = C.end()`

Iterator, der auf das Element *hinter dem letzten Element* zeigt³

- `*it`

²Sehr nützlich für unhandliche Wiedergabetypen

³PTE: Past-the-End

Iteratoren auf Containern verwenden

Sehr einfach und absichtlich immer gleich!

Gegeben sei ein Container namens `C`

- `auto2 it = C.begin()`

Iterator, der auf das erste Element zeigt

- `auto it = C.end()`

Iterator, der auf das Element *hinter dem letzten Element* zeigt³

- `*it`

Zugriff (und eventuell Änderung) auf das aktuelle Element

- `++it`

²Sehr nützlich für unhandliche Wiedergabetypen

³PTE: Past-the-End

Iteratoren auf Containern verwenden

Sehr einfach und absichtlich immer gleich!

Gegeben sei ein Container namens `C`

- `auto2 it = C.begin()`

Iterator, der auf das erste Element zeigt

- `auto it = C.end()`

Iterator, der auf das Element *hinter dem letzten Element* zeigt³

- `*it`

Zugriff (und eventuell Änderung) auf das aktuelle Element

- `++it`

Iterator um ein Element weitersetzen



²Sehr nützlich für unhandliche Wiedergabetypen

³PTE: Past-the-End

8. Aufgabe "Find Max"

Aufgabe "Find Max"

Aufgabe "Find Max"

```
// PRE: i < j <= v.size()
// POST: Returns the greatest element of all elements
//        with indices between i and j (excluding j)
int find_max(const std::vector<int>& v, int i, int j) {
    int max_value = 0;

    for (; i < j; ++i) {
        if (max_value < v[i]) {
            max_value = v[i];
        }
    }

    return max_value;
}
```

Aufgabe "Find Max"

Aufgabe "Find Max"

- Öffnet "Find Max" auf **code expert**

Aufgabe "Find Max"

- Öffnet "Find Max" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet

Aufgabe "Find Max"

- Öffnet "Find Max" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet
- Programmiert eine Lösung (optional in Gruppen)

Aufgabe "Find Max" (Lösung)

Aufgabe "Find Max" (Lösung)

```
// PRE: (begin < end) && (begin and end must be valid iterators)
// POST: Return the greatest element in the range [begin, end)
int find_max(std::vector<int>::iterator begin,
             std::vector<int>::iterator end) {
    int max_value = 0;

    for(; begin != end; ++begin) {
        if (max_value < *begin) {
            max_value = *begin;
        }
    }

    return max_value;
}
```

Fragen/Unklarheiten?

Die algorithm Library

- Sicherlich hat jemand Schlaueres bereits alle gängigen Algorithmen für uns implementiert, oder?

Die `algorithm` Library

- Sicherlich hat jemand Schlaueres bereits alle gängigen Algorithmen für uns implementiert, oder?
- Ja! Die `algorithm`-Library!

Die `algorithm` Library

- Sicherlich hat jemand Schlaueres bereits alle gängigen Algorithmen für uns implementiert, oder?
- Ja! Die `algorithm`-Library!
- Diese Funktionen sind so konzipiert, dass sie mit verschiedenen Containern wie Vektoren, Arrays, Listen usw. arbeiten und dabei helfen, Aufgaben effizient auszuführen, ohne dass die Algorithmen jedes mal von Grund auf neu geschrieben werden müssen

Die `algorithm` Library

- Sicherlich hat jemand Schläueres bereits alle gängigen Algorithmen für uns implementiert, oder?
- Ja! Die `algorithm`-Library!
- Diese Funktionen sind so konzipiert, dass sie mit verschiedenen Containern wie Vektoren, Arrays, Listen usw. arbeiten und dabei helfen, Aufgaben effizient auszuführen, ohne dass die Algorithmen jedes mal von Grund auf neu geschrieben werden müssen
- Nicht vergessen: `#include <algorithm>`

Aufgabe "The algorithm Library"

Aufgabe "The algorithm Library"

- Öffnet "The algorithm Library" auf **code expert**

Aufgabe "The algorithm Library"

- Öffnet "The algorithm Library" auf **code expert**
- Überlegt euch, wie ihr das Problem angehen würdet

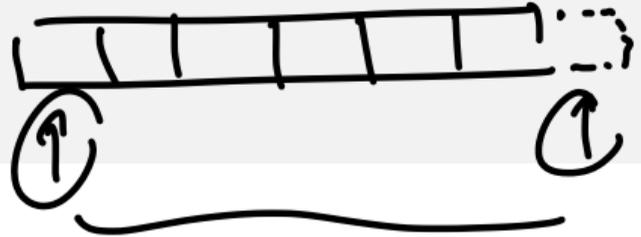
Aufgabe "The algorithm Library"

- Öffnet "The algorithm Library" auf **code expert**
- Überlegt euch, wie ihr das Problem angehen würdet
- Programmiert eine Lösung (optional in Gruppen)

Aufgabe "The algorithm Library" (Lösung)

Aufgabe "The algorithm Library" (Lösung)

```
// ...  
int largest_element = alg lib. *std::max_element(vec.begin(), vec.end());  
// ...  
std::sort(vec.begin(), vec.end());  
// ...
```



Fragen/Unklarheiten?

9. Rekursion

Aufgabe "Recursion to Iteration 1"

Aufgabe "Recursion to Iteration 1"

- Öffnet "Recursion to Iteration 1" auf **code expert**

Aufgabe "Recursion to Iteration 1"

- Öffnet "Recursion to Iteration 1" auf **code expert**
- Überlegt euch, wie ihr das Problem angehen würdet

Aufgabe "Recursion to Iteration 1"

- Öffnet "Recursion to Iteration 1" auf **code expert**
- Überlegt euch, wie ihr das Problem angehen würdet
- Programmiert eine Lösung (optional in Gruppen)

Aufgabe "Recursion to Iteration 1" (Lösung)

Aufgabe "Recursion to Iteration 1" (Lösung)

```
// PRE: n >= 0
int f_it(const int n) {
    if (n <= 2) {
        return 1;
    }
    int a = 1;           // f(0)
    int b = 1;           // f(1)
    int c = 1;           // f(2)
    for (int i = 3; i < n; ++i) {
        const int a_prev = a; // f(i-3)
        a = b;               // f(i-2)
        b = c;               // f(i-1)
        c = b + 2 * a_prev;  // f(i)
    }
    return c + 2 * a;      // f(n-1) + 2 * f(n-3)
}
```

Aufgabe "Recursion to Iteration 2"

Aufgabe "Recursion to Iteration 2"

- Öffnet "Recursion to Iteration 2" auf **code expert**

Aufgabe "Recursion to Iteration 2"

- Öffnet "Recursion to Iteration 2" auf **code expert**
- Überlegt euch, wie ihr das Problem angehen würdet

Aufgabe "Recursion to Iteration 2"

- Öffnet "Recursion to Iteration 2" auf **code expert**
- Überlegt euch, wie ihr das Problem angehen würdet
- Programmiert eine Lösung (optional in Gruppen)

Aufgabe "Recursion to Iteration 2" (Lösung)

Aufgabe "Recursion to Iteration 2" (Lösung)

```
// PRE: n >= 0
int f_it(const int n) {
    if (n == 0) { // special case
        return 1;
    }

    std::vector<int> f_values(n+1, 0);
    f_values[0] = 1;

    for (int i = 1; i <= n; ++i) {
        f_values[i] = f_values[i-1] + 2 * f_values[i / 2];
    }

    return f_values[n];
}
```

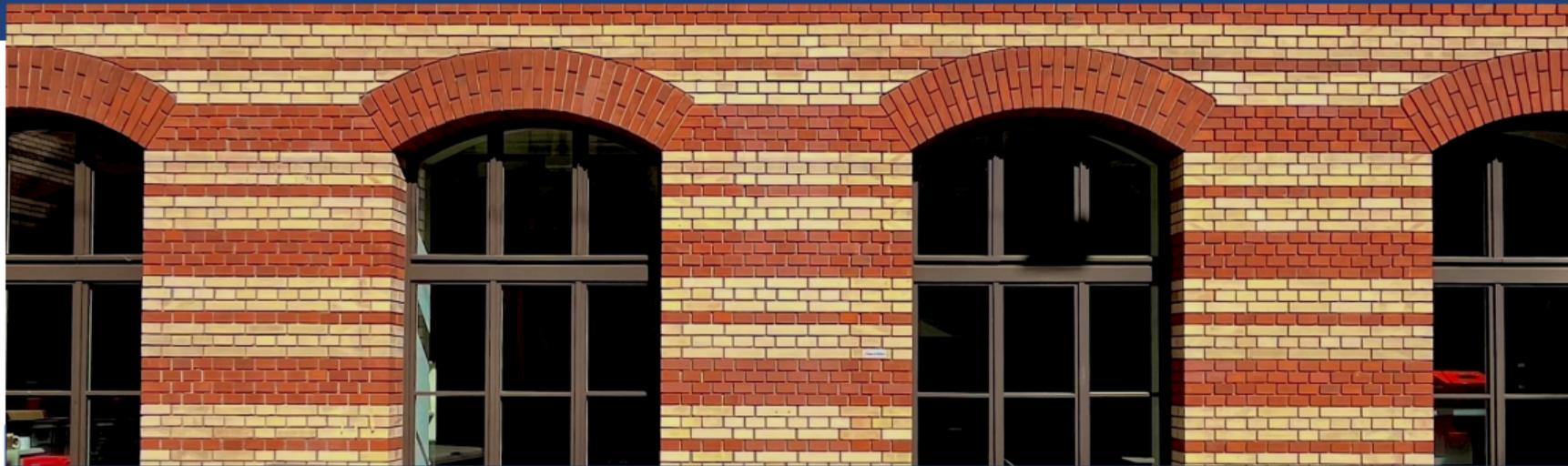
Fragen/Unklarheiten?

10. Outro

Allgemeine Fragen?

Bis zum nächsten Mal

Schöne Woche noch!



Übungsstunde — Informatik — 10

Adel Gavranović

Pointer-relevante Operatoren, Referenzen vs. Pointer, Iteratoren, `this->`, dynamischer Speicher

Übersicht

Follow-up

& VS *

Referenzen vs Pointer

this->

Dynamische Datenstrukturen & Iteratoren

Our_list Grundmaterial

Our_list Bonusmaterial



n.ethz.ch/~agavranovic

 Material

 Webpage

 Mail

1. Follow-up

Anzahl Argumente für überladene Operatoren

Anzahl Argumente für überladene Operatoren

Letzte Übungsstunde was ich überrascht, dass overloaded operators nur einen Input benötigen. Jetzt weiss ich wieso!

Anzahl Argumente für überladene Operatoren

Letzte Übungsstunde was ich überrascht, dass overloaded operators nur einen Input benötigen. Jetzt weiss ich wieso!

```
Tribool Tribool::operator&&(const Tribool& other) const {  
    Tribool result(std::min(value, other.value));  
    return result;  
}
```

A ~~aa~~ B

Anzahl Argumente für überladene Operatoren

Letzte Übungsstunde was ich überrascht, dass overloaded operators nur einen Input benötigen. Jetzt weiss ich wieso!

```
Tribool Tribool::operator&&(const Tribool& other) const {  
    Tribool result(std::min(value, other.value));  
    return result;  
}
```

Folgendes ist equivalent:

```
Tribool A("True"), B("Unknown");  
Tribool C = A && B;           // infix notation  
// Equivalent to  
Tribool D = A.operator&&(B);  // explicit member function notation
```

Non-member Funktionen brauchen noch immer beide Objekte als Input!

Ranged for-loops

Ranged for-loops

Letzte Übungsstunde habe ich einen Fehler beim Erklären der "Ranged for-loops" gemacht — Bitte entschuldigt die Verwirrung.

Ranged for-loops

Letzte Übungsstunde habe ich einen Fehler beim Erklären der "Ranged for-loops" gemacht — Bitte entschuldigt die Verwirrung.

```
std::vector<int> numbers = {1, 2, 3, 4, 5};

for (int i : numbers) {
    std::cout << (i += 1) << " ";
}

std::cout << "| ";

for (int i : numbers) {
    std::cout << i << " ";
}
```

Ranged for-loops

Letzte Übungsstunde habe ich einen Fehler beim Erklären der "Ranged for-loops" gemacht — Bitte entschuldigt die Verwirrung.

```
std::vector<int> numbers = {1, 2, 3, 4, 5};

for (int i : numbers) {
    std::cout << (i += 1) << " ";
}

std::cout << "| ";

for (int i : numbers) {
    std::cout << i << " ";
}
```

Was wird hier der Output sein?

Ranged for-loops

Letzte Übungsstunde habe ich einen Fehler beim Erklären der "Ranged for-loops" gemacht — Bitte entschuldigt die Verwirrung.

```
std::vector<int> numbers = {1, 2, 3, 4, 5};

for (int i : numbers) {
    std::cout << (i += 1) << " ";
}

std::cout << "| ";

for (int i : numbers) {
    std::cout << i << " ";
}
```

Was wird hier der Output sein? 2 3 4 5 6 | 1 2 3 4 5

Ranged for-loops mit Referenzen

Ranged for-loops mit Referenzen

Die Elemente im Container werden also "direkt" verfügbar gemacht, also keine Iteratoren nötig!

Ranged for-loops mit Referenzen

Die Elemente im Container werden also "direkt" verfügbar gemacht, also keine Iteratoren nötig!

```
std::vector<int> numbers = {1, 2, 3, 4, 5};

for (int& i : numbers) {                               // referenced (&) this time!
    std::cout << (i += 1) << " ";
}

std::cout << "| ";

for (int i : numbers) {
    std::cout << i << " ";
}
```

Ranged for-loops mit Referenzen

Die Elemente im Container werden also "direkt" verfügbar gemacht, also keine Iteratoren nötig!

```
std::vector<int> numbers = {1, 2, 3, 4, 5};

for (int& i : numbers) {                               // referenced (&) this time!
    std::cout << (i += 1) << " ";
}

std::cout << "| ";

for (int i : numbers) {
    std::cout << i << " ";
}
```

Was wird hier der Output sein?

Ranged for-loops mit Referenzen

Die Elemente im Container werden also "direkt" verfügbar gemacht, also keine Iteratoren nötig!

```
std::vector<int> numbers = {1, 2, 3, 4, 5};

for (int& i : numbers) {                               // referenced (&) this time!
    std::cout << (i += 1) << " ";
}

std::cout << "| ";

for (int i : numbers) {
    std::cout << i << " ";
}
```

Was wird hier der Output sein? 2 3 4 5 6 | 2 3 4 5 6

Fragen/Unklarheiten?

2. Feedback zu **code** expert

Allgemeines bezüglich **code expert**

Aufgabe "Recursive Function Analysis"

Aufgabe "Recursive Function Analysis"

- Achtet auf die Details!

Musterlösung für Teil 1

Musterlösung

```
bool f(const int n) {  
    if (n == 0) return false;  
    return !f(n - 1);  
}
```

Musterlösung für Teil 1

Musterlösung

```
bool f(const int n) {  
    if (n == 0) return false;  
    return !f(n - 1);  
}
```

i) |

```
// PRE:  n >= 0  
// POST: returns `false` if n is even  
//       returns `true`  if n is odd
```

Musterlösung für Teil 1

Musterlösung

```
bool f(const int n) {  
    if (n == 0) return false;  
    return !f(n - 1);  
}
```

i) |

```
// PRE:  n >= 0  
// POST: returns `false` if n is even  
//       returns `true`  if n is odd
```

ii) The function `f` immediately terminates for `n == 0`. With each recursive call `n` is decremented, i.e., `f` is called with parameter `<n`, eventually reaching `n == 0` (at which point it terminates — as mentioned above).

Musterlösung für Teil 1

Musterlösung

```
bool f(const int n) {  
    if (n == 0) return false;  
    return !f(n - 1);  
}
```

i) |

```
// PRE:  n >= 0  
// POST: returns `false` if n is even  
//       returns `true`  if n is odd
```

- ii) The function `f` immediately terminates for `n == 0`. With each recursive call `n` is decremented, i.e., `f` is called with parameter `<n`, eventually reaching `n == 0` (at which point it terminates — as mentioned above).
- iii) $\text{Calls}_f(n) = n + 1$ (including first non-rec. call)

Musterlösung für Teil 2

Musterlösung

```
void g(const int n) {  
    if (n == 0) {  
        std::cout << "*"; return;  
    }  
    g(n - 1); g(n - 1);  
}
```

Musterlösung für Teil 2

Musterlösung

```
void g(const int n) {  
    if (n == 0) {  
        std::cout << "*"; return;  
    }  
    g(n - 1); g(n - 1);  
}
```

i) | `// PRE: n >= 0`
| `// POST: prints 2n stars to std::out`

Musterlösung für Teil 2

Musterlösung

```
void g(const int n) {  
    if (n == 0) {  
        std::cout << "*"; return;  
    }  
    g(n - 1); g(n - 1);  
}
```

i) | `// PRE: n >= 0`
| `// POST: prints 2n stars to std::out`

- ii) The function `g` immediately terminates for `n == 0`. With each recursive call `n` is decremented, i.e., `g` is called with parameter `< n`, eventually reaching `n == 0`. This is true for both recursive calls of `g`.

Musterlösung für Teil 2

Musterlösung

```
void g(const int n) {  
    if (n == 0) {  
        std::cout << "*"; return;  
    }  
    g(n - 1); g(n - 1);  
}
```

i) | `// PRE: n >= 0`
| `// POST: prints 2^n stars to std::out`

- ii) The function `g` immediately terminates for `n == 0`. With each recursive call `n` is decremented, i.e., `g` is called with parameter `< n`, eventually reaching `n == 0`. This is true for both recursive calls of `g`.
- iii) $\text{Calls}_g(n) = \sum_{i=0}^n 2^i = 2^{n+1} - 1$
(including first non-rec. call)

Fragen bezüglich **code expert** eurerseits?

3. Lernziele

Ziele

- Unterschiede zwischen Pointern und Referenzen verstehen
- Programme mit Pointern tracen und schreiben können
- Programme mit dynamischem Speicher schreiben können
- Simple Container implementieren können

4. Zusammenfassung

5. & VS *

Bedeutungen von &

Das Symbol & hat in C++ viele Bedeutungen. Das ist verwirrend. Es hat 3 *verschiedene Bedeutungen*, je nach Position im Code:

Bedeutungen von &

Das Symbol & hat in C++ viele Bedeutungen. Das ist verwirrend. Es hat 3 *verschiedene Bedeutungen*, je nach Position im Code:

Bedeutung von &

Bedeutungen von &

Das Symbol & hat in C++ viele Bedeutungen. Das ist verwirrend. Es hat 3 *verschiedene Bedeutungen*, je nach Position im Code:

Bedeutung von &

1. als AND-operator

```
bool z = x && y;
```

Bedeutungen von &

Das Symbol & hat in C++ viele Bedeutungen. Das ist verwirrend. Es hat 3 *verschiedene Bedeutungen*, je nach Position im Code:

Bedeutung von &

1. als AND-operator

```
bool z = x && y;
```

2. um eine Variable als Alias zu deklarieren

```
int& y = x;
```

Bedeutungen von &

Das Symbol & hat in C++ viele Bedeutungen. Das ist verwirrend. Es hat 3 *verschiedene Bedeutungen*, je nach Position im Code:

Bedeutung von &

1. als AND-operator

```
bool z = x && y;
```

2. um eine Variable als Alias zu deklarieren

```
int& y = x;
```

3. um die Adresse einer Variable zu erhalten (address-operator)

```
int *ptr_a = &a;
```

Bedeutungen von *

Dito mit dem Symbol *.

Bedeutung von *

Bedeutungen von *

Dito mit dem Symbol *.

Bedeutung von *

1. als (arithmetischer) Multiplikation-operator

$$z = x * y;$$

Bedeutungen von *

Dito mit dem Symbol *.

Bedeutung von *

1. als (arithmetischer) Multiplikation-operator

`z = x * y;`

2. um eine Pointer-Variable zu deklarieren

`int* ptr_a = &a;`



Bedeutungen von *

Dito mit dem Symbol *.

Bedeutung von *

1. als (arithmetischer) Multiplikation-operator

```
z = x * y;
```

2. um eine Pointer-Variable zu deklarieren

```
int* ptr_a = &a;
```

3. um auf eine Variable via ihrem Pointer zuzugreifen
(dereference-operator)

```
int a = *ptr_a;
```

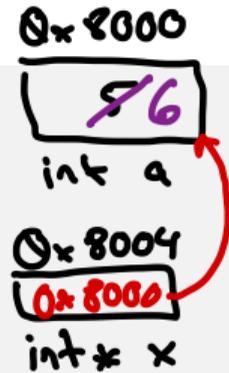
Fragen/Unklarheiten?

6. Referenzen vs Pointer

Pointer Basics

Versucht, folgendes Programm¹ detailliert zu tracen

```
int main() {  
  
    int a = 5;  
    int* x = &a;  
    (*x) = 6;  
    a  
    return 0;  
}
```



¹Vollständiger Trace [hier](#)

References

```
void references(){  
    int a = 1;  
    int b = 2;  
    int& x = a;  
    int& y = x;  
    y = b;  
  
    std::cout  
    << a << " "  
    << b << " "  
    << x << " "  
    << y << std::endl;  
}
```

Trace das Programm und schreibe den erwarteten Output hin, wenn die Funktion aufgerufen wurde

References

```
void references(){  
    int a = 1;  
    int b = 2;  
    int& x = a;  
    int& y = x;  
    y = b;  
  
    std::cout  
    << a << " "  
    << b << " "  
    << x << " "  
    << y << std::endl;  
}
```

Trace das Programm und schreibe den erwarteten Output hin, wenn die Funktion aufgerufen wurde

2 2 2 2

Pointers

```
void pointers(){  
    int a = 1;  
    int b = 2;  
    int* x = &a;  
    int* y = x;  
  
    std::cout  
    << a << " "  
    << b << " "  
    << x << " "  
    << y << std::endl;  
}
```

Trace das Programm und schreibe den erwarteten Output hin, wenn die Funktion aufgerufen wurde

Pointers

```
void pointers(){  
    int a = 1;  
    int b = 2;  
    int* x = &a;  
    int* y = x;  
  
    std::cout  
    << a << " "  
    << b << " "  
    << x << " "  
    << y << std::endl;  
}
```

Trace das Programm und schreibe den erwarteten Output hin, wenn die Funktion aufgerufen wurde

1 2 0x7ffe4d1fb904 0x7ffe4d1fb904

Pointers

```
void pointers(){  
    int a = 1;  
    int b = 2;  
    int* x = &a;  
    int* y = x;  
  
    std::cout  
    << a << " "  
    << b << " "  
    << x << " "  
    << y << std::endl;  
}
```

Trace das Programm und schreibe den erwarteten Output hin, wenn die Funktion aufgerufen wurde

1 2 0x7ffe4d1fb904 0x7ffe4d1fb904

(Die Adressen könnten bei jedem Aufruf anders sein!)

Pointers and Addresses

```
void ptrs_and_addresses(){
    int a = 5;
    int b = 7;

    int* x = nullptr;
    x = &a;

    std::cout << a << "\n";
    std::cout << *x << "\n";

    std::cout << x << "\n";
    std::cout << &a << "\n";
}
```

Trace das Programm und schreibe den erwarteten Output hin, wenn die Funktion aufgerufen wurde

Pointers and Addresses

```
void ptrs_and_addresses(){
    int a = 5;
    int b = 7;

    int* x = nullptr;
    x = &a;

    std::cout << a << "\n";
    std::cout << *x << "\n";

    std::cout << x << "\n";
    std::cout << &a << "\n";
}
```

Trace das Programm und schreibe den erwarteten Output hin, wenn die Funktion aufgerufen wurde

5

5

0x7ffe4d1fb914

0x7ffe4d1fb914

(Die Adressen könnten bei jedem Aufruf anders sein!)

Fragen/Unklarheiten?

7. this->

Was zum f*&k ist this->?

Bedeutung von this->

this-> hat zwei Teile

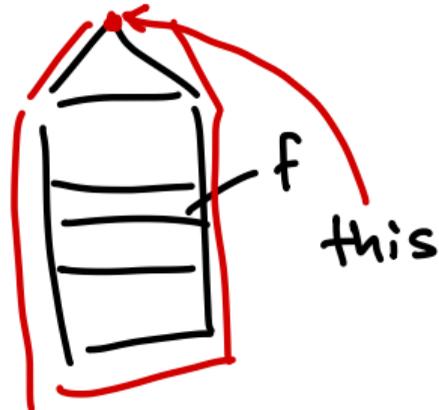
Was zum f*&k ist this->?

Bedeutung von this->

this-> hat zwei Teile

- this

- ist ein Pointer zum *aktuellen* Objekt (Class oder Struct T)



Was zum f*&k ist this->?

Bedeutung von this->

this-> hat zwei Teile

- **this**

- ist ein Pointer zum *aktuellen* Objekt (Class oder Struct T)
- also vom Typ T*

Was zum f*&k ist this->?

Bedeutung von this->

this-> hat zwei Teile

- **this**

- ist ein Pointer zum *aktuellen* Objekt (Class oder Struct T)
- also vom Typ T*

- **->**

- ist ein sehr cool aussehender Operator

Was zum f*&k ist this->?

Bedeutung von this->

this-> hat zwei Teile

- this

- ist ein Pointer zum *aktuellen* Objekt (Class oder Struct T)
- also vom Typ T*

- ->

- ist ein sehr cool aussehender Operator
- `this->member_element` ist äquivalent zu `*(this).member_element`
- Der Pfeil-Operator dereferenziert einen Pointer zu einem Objekt, um auf einen seiner Members zuzugreifen (Funktionen oder Variablen)

this → next → next ;
≙ * (* (*this).next).next

Beispiel

Wofür wird `this` hier benutzt?

```
struct WeirdNumber {  
  
    int number;  
  
    void increment_by(int number){  
        (*this).number = (*this).number + number;  
        // or  
        // this->number = this->number + number;  
    }  
};
```

Beispiel

Wofür wird **this** hier benutzt?

```
struct WeirdNumber {  
  
    int number;  
  
    void increment_by(int number){  
        (*this).number = (*this).number + number;  
        // or  
        // this->number = this->number + number;  
    }  
};
```

Um die beiden gleichnamigen Variablen `number` zu unterscheiden

Beispiel

```
int main(){

    WeirdNumber a = {42};
    WeirdNumber b = {-17};

    a.increment_by(3);
    // 'this' in the call of the increment_by function
    // refers to the object a.
    b.increment_by(2);
    // 'this' in the call of the increment_by function
    // refers to the object b.

    std::cout << a.number << " " << b.number << std::endl;

    return 0;
}
```

8. Dynamische Datenstrukturen & Iteratoren

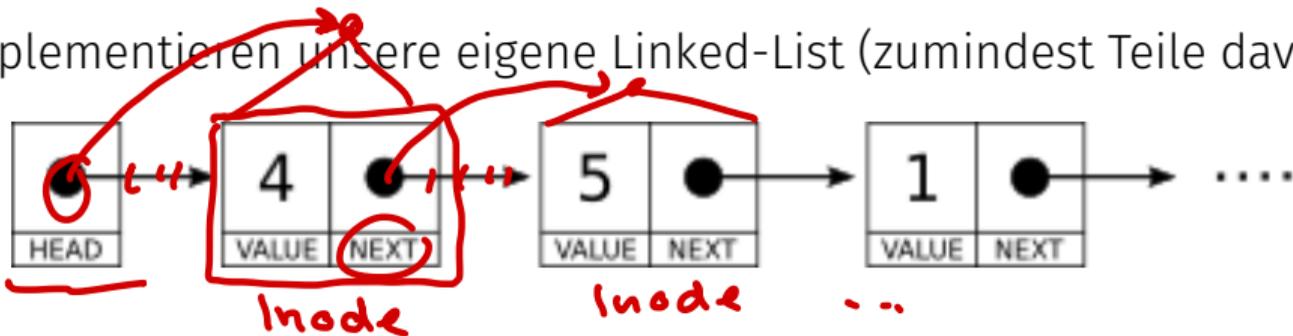
8. Dynamische Datenstrukturen & Iteratoren

8.1. `Our_list` Grundmaterial

Wir implementieren unsere eigene Linked-List (zumindest Teile davon)

our_list

Wir implementieren unsere eigene Linked-List (zumindest Teile davon)

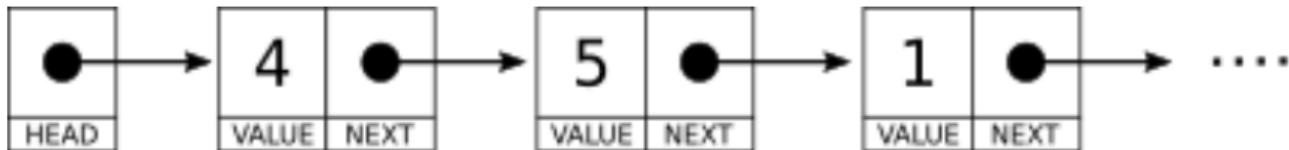


- Eine Liste besteht aus "Blöcken" von `lnodes`, wobei eine `lnode` immer auf die nächste zeigt



our_list

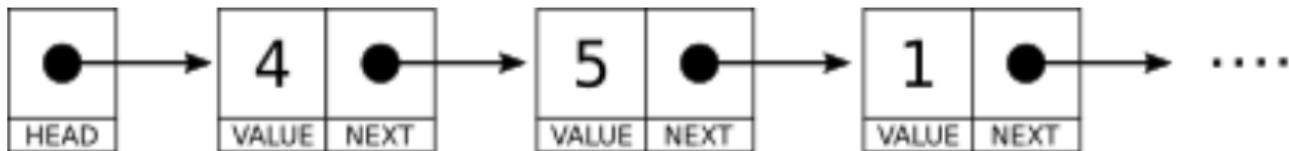
Wir implementieren unsere eigene Linked-List (zumindest Teile davon)



- Eine Liste besteht aus "Blöcken" von `lnodes`, wobei eine `lnode` immer auf die nächste zeigt
- Aber was ist überhaupt eine `lnode`?

our_list

Wir implementieren unsere eigene Linked-List (zumindest Teile davon)



- Eine Liste besteht aus "Blöcken" von `lnodes`, wobei eine `lnode` immer auf die nächste zeigt
- Aber was ist überhaupt eine `lnode`?
- Antwort: ein Struct, das aus einem `int` `value` und einem `lnode` pointer besteht

our_list

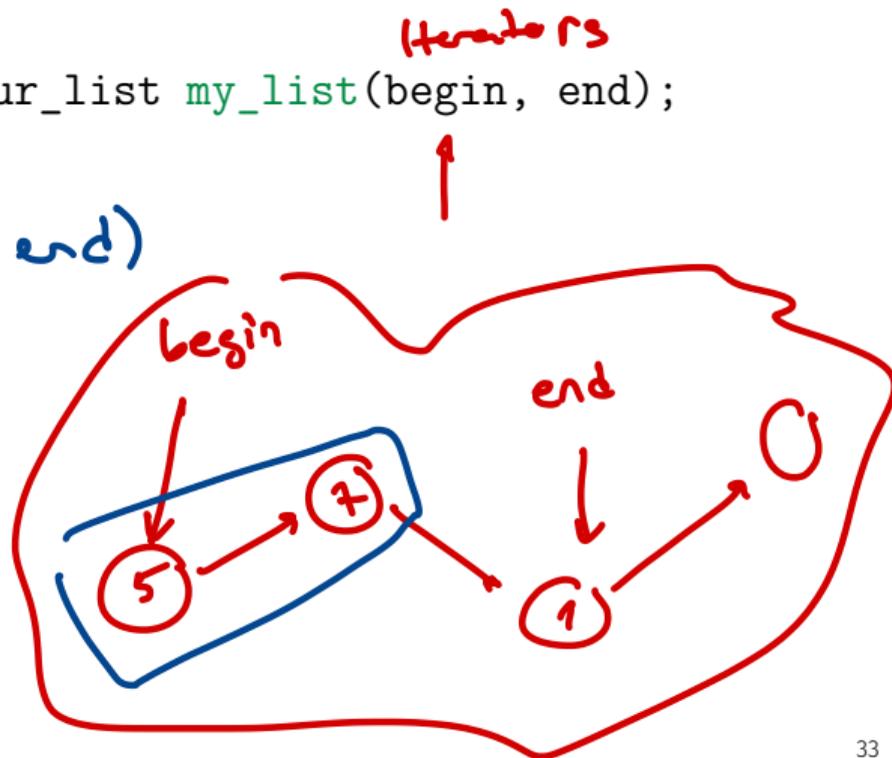
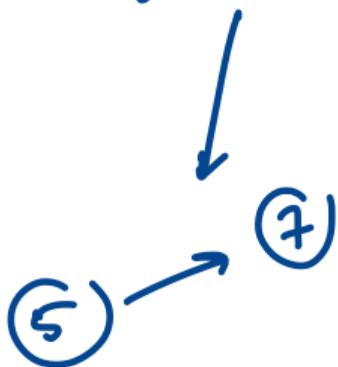
Erste Aufgabe: Implementiere einen Constructor, der eine neue Liste mit Iteratoren initialisiert

our_list

Erste Aufgabe: Implementiere einen Constructor, der eine neue Liste mit Iteratoren initialisiert

- Wir wollen schreiben können: `our_list my_list(begin, end);`

our_list my_list(begin, end)



Erste Aufgabe: Implementiere einen Constructor, der eine neue Liste mit Iteratoren initialisiert

- Wir wollen schreiben können: `our_list my_list(begin, end);`
- Idee: Benutze die Iteratoren, um neue `lnodes` in die Liste hinzuzufügen

Erste Aufgabe: Implementiere einen Constructor, der eine neue Liste mit Iteratoren initialisiert

- Wir wollen schreiben können: `our_list my_list(begin, end);`
- Idee: Benutze die Iteratoren, um neue `lnodes` in die Liste hinzuzufügen
- Wie können wir auf die verschiedenen Elemente zugreifen?

Erste Aufgabe: Implementiere einen Constructor, der eine neue Liste mit Iteratoren initialisiert

- Wir wollen schreiben können: `our_list my_list(begin, end);`
- Idee: Benutze die Iteratoren, um neue `lnodes` in die Liste hinzuzufügen
- Wie können wir auf die verschiedenen Elemente zugreifen?
 - Zugriff auf Wert der `lnode`, auf die der Iterator zeigt:

`*it`

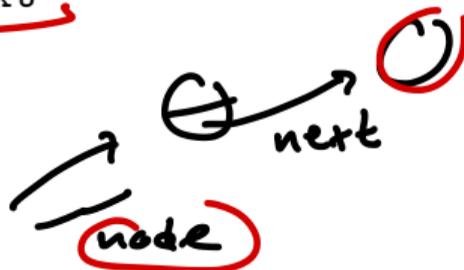
Erste Aufgabe: Implementiere einen Constructor, der eine neue Liste mit Iteratoren initialisiert

- Wir wollen schreiben können: `our_list my_list(begin, end);`
- Idee: Benutze die Iteratoren, um neue `lnodes` in die Liste hinzuzufügen
- Wie können wir auf die verschiedenen Elemente zugreifen?
 - Zugriff auf Wert der `lnode`, auf die der Iterator zeigt:

`*it`

- Nächste `lnode` in der Folge:

`node` → `next`



Erste Aufgabe: Implementiere einen Constructor, der eine neue Liste mit Iteratoren initialisiert

- Wir wollen schreiben können: `our_list my_list(begin, end);`
- Idee: Benutze die Iteratoren, um neue `lnodes` in die Liste hinzuzufügen
- Wie können wir auf die verschiedenen Elemente zugreifen?

- Zugriff auf Wert der `lnode`, auf die der Iterator zeigt:

`*it`

- Nächste `lnode` in der Folge:

`node->next`

- Pointer zu neuer `lnode` erstellen::

`new lnode{value, pointer}`

Denkt daran: `new T` gibt einen `T*` zurück

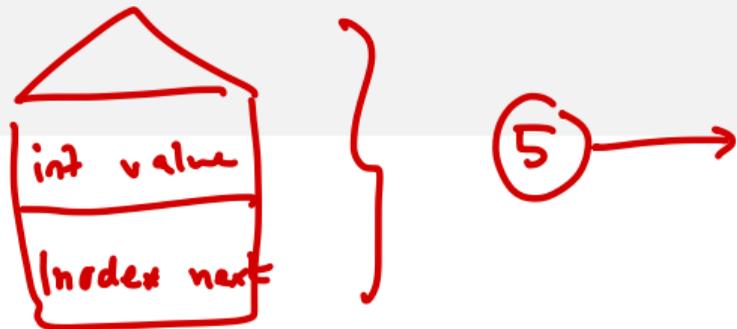
our_list: class our_list

```
class our_list {  
    struct lnode {  
        // ...  
    };  
  
    lnode* head;  
  
public:  
  
    class const_iterator {  
        // ...  
    };  
  
    // member functions  
};
```

HEAD
lnode*

our_list: struct lnode

```
//                               in class our_list                               //  
struct lnode {  
    int value;  
    lnode* next;  
};
```



our_list: const_iterator

```
//                               in class our_list                               //
class const_iterator {
    const lnode* node;
public:
    const_iterator(const lnode* const n);
    // PRE: Iterator doesn't point to the element beyond the last one
    // POST: Iterator points to the next element
    const_iterator& operator++(); // Pre-increment
    // POST: Return the reference to the number at which the
    //       iterator is currently pointing
    const int& operator*() const;
    // True if iterators are pointing to different elements
    bool operator!=(const const_iterator& other) const;
    // True if iterators are pointing to the same element
    bool operator==(const const_iterator& other) const;
};
```

our_list: Memberfunktionen

```
//          in class our_list          //  
our_list();  
  
// PRE: begin and end are iterators pointing to the same vector  
//      and begin is before end  
// POST: Constructed our_list contains all elements between begin and end  
our_list(const_iterator begin, const_iterator end);  
  
// POST: e is appended at the beginning of the vector  
void push_front(int e);  
  
// POST: Returns an iterator that points to the first element  
const_iterator begin() const;  
  
// POST: Returns an iterator that points after the last element  
const_iterator end() const;
```



L.push_front(5);
↓



Aufgabe "our_list::init"

Aufgabe "our_list::init"

- Öffnet "our_list::init" auf **code expert**

Aufgabe "our_list::init"

- Öffnet "our_list::init" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet

Aufgabe "our_list::init"

- Öffnet "our_list::init" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet
- Programmiert eine Lösung (optional in Gruppen)

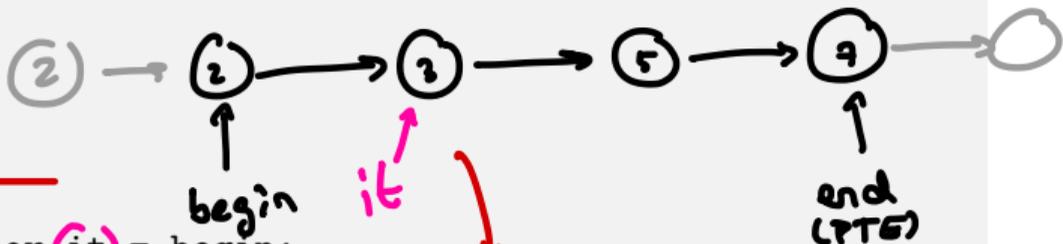
Aufgabe "our_list::init" (Lösung)

Aufgabe "our_list::init" (Lösung)

in ...cpp

```
xxx our_list::our_list(our_list::const_iterator begin, our_list::const_iterator end) {
```

```
    this->head = nullptr;
    if (begin == end) {
        return;
    }
```



```
    // add first element
```

```
    our_list::const_iterator it = begin;
```

```
    this->head = new lnode { *it, nullptr };
    ++it;
```

```
    lnode *node = this->head;
```

```
    // add remaining elements
```

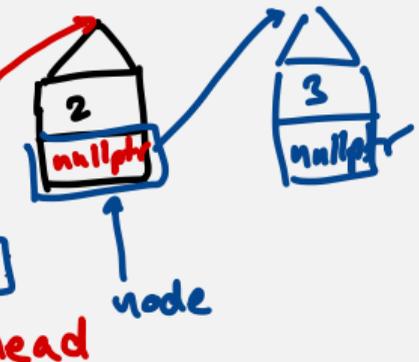
```
    for (; it != end; ++it) {
```

```
        node->next = new lnode { *it, nullptr };
        node = node->next;
```

```
    }
```

```
}
```

leaf
lindo
case
auto



Fragen/Unklarheiten?

Zweite Aufgabe: Implementiere eine Funktion der Class "our_list", die eine Node mit der nächsten tauscht

Zweite Aufgabe: Implementiere eine Funktion der Class "our_list", die eine Node mit der nächsten tauscht

- Ihr könnt eine recht ähnliche Herangehensweise wie bei anderen Swap-Funktionen benutzen (also mit einer temporären Variable `tmp`)

Zweite Aufgabe: Implementiere eine Funktion der Class "our_list", die eine Node mit der nächsten tauscht

- Ihr könnt eine recht ähnliche Herangehensweise wie bei anderen Swap-Funktionen benutzen (also mit einer temporären Variable `tmp`)
- Aber:
 - Benutzt Pointer
 - Was passiert im Fall "0" (wenn der Head Pointer getauscht werden soll)?
 - Wie könnt ihr vermeiden, dass nicht plötzlich auf unerlaubten Speicher zugegriffen wird?

Aufgabe "our_list::swap"

Aufgabe "our_list::swap"

- Öffnet "our_list::swap" auf **code expert**

Aufgabe "our_list::swap"

- Öffnet "our_list::swap" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet

Aufgabe "our_list::swap"

- Öffnet "our_list::swap" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet
- Programmiert eine Lösung (optional in Gruppen)

Aufgabe "our_list::swap" (Lösung)

Aufgabe "our_list::swap" (Lösung)

```
void our_list::swap(int index) {  
  
    if (index == 0) {  
  
        assert(this->head != nullptr);  
        assert(this->head->next != nullptr);  
  
        lnode* tmp = this->head->next;  
        this->head->next = this->head->next->next;  
        tmp->next = this->head;  
        this->head = tmp;  
  
    } else { /* ... */ }
```

Aufgabe "our_list::swap" (Lösung)

```
else { lnode* prev = nullptr;
      lnode* curr = this->head;

      while (index > 0) { // Find the element
        prev = curr;
        curr = curr->next;
        --index;
      }

      assert(curr != nullptr);
      assert(curr->next != nullptr);

      lnode* tmp = curr->next; // Swap with the next one
      curr->next = curr->next->next;
      tmp->next = curr;
      prev->next = tmp;
    }}// two '}' to close function
```

Fragen/Unklarheiten?

8. Dynamische Datenstrukturen & Iteratoren

8.2. Our_list Bonusmaterial

Aufgabe "our_list::extend"

Aufgabe "our_list::extend"

- Öffnet "our_list::extend" auf **code expert**

Aufgabe "our_list::extend"

- Öffnet "our_list::extend" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet

Aufgabe "our_list::extend"

- Öffnet "our_list::extend" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet
- Programmiert eine Lösung (optional in Gruppen)

Aufgabe "our_list::extend" (Lösung)

Aufgabe "our_list::extend" (Lösung)

```
void our_list::extend(our_list::const_iterator begin,
                    our_list::const_iterator end) {
    if (begin == end) { return; }
    our_list::const_iterator it = begin;
    if (this->head == nullptr) {
        this->head = new lnode { *it, nullptr };
        ++it;
    }
    lnode *n = this->head;
    while (n->next != nullptr) {
        n = n->next;
    }
    for (; it != end; ++it) {
        n->next = new lnode { *it, nullptr };
        n = n->next;
    }
}
```

Fragen/Unklarheiten?

Aufgabe "our_list::merge_sorted" (Schwierig)

Aufgabe "our_list::merge_sorted" (Schwierig)

Falls all diese Klassen, Pointers und dynamischen Datenzuweisungen nicht schon schwierig genug für euch waren, lasst uns auch noch Rekursion dazunehmen!

Merge-Sort

Merge-Sort

- Goal: Sort an **arbitrary** array as **quickly** as possible.

5	2	8	7	7	3	1
---	---	---	---	---	---	---



1	2	3	5	7	7	8
---	---	---	---	---	---	---

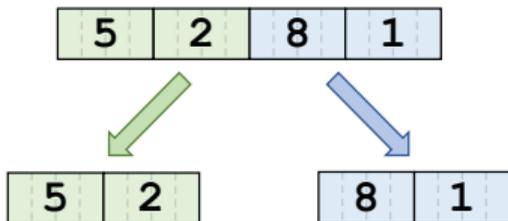
Merge-Sort

- Idea: **Divide and Conquer**

Merge-Sort

- Idea: **Divide and Conquer**

1. Split whole array into two parts. (**Divide**)

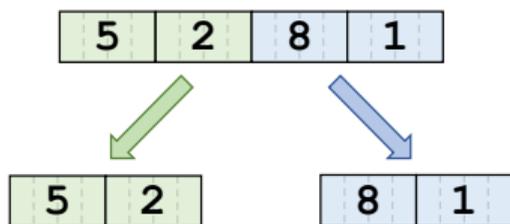


Divide

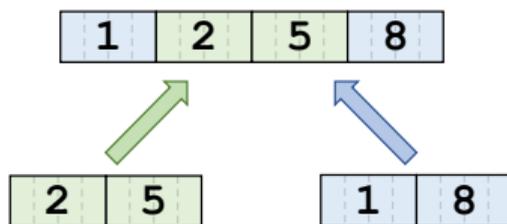
Merge-Sort

- Idea: **Divide and Conquer**

1. Split whole array into two parts. (**Divide**)
2. Then sort these and combine them. (**Conquer**)



Divide

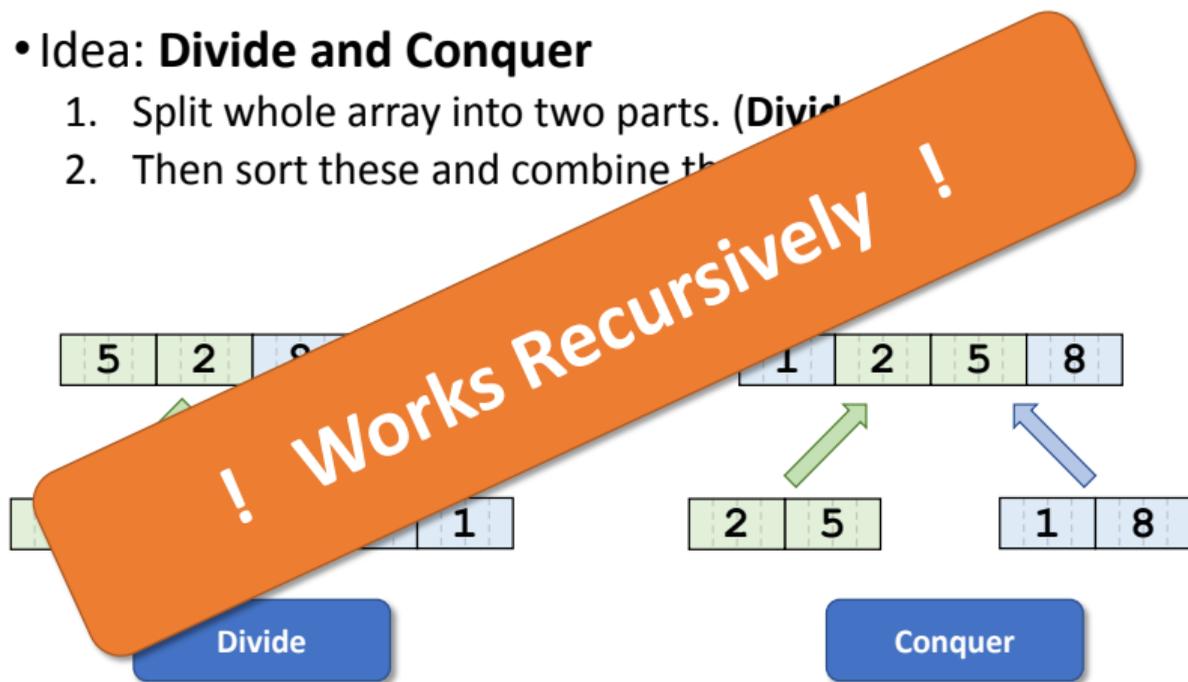


Conquer

Merge-Sort

- Idea: **Divide and Conquer**

1. Split whole array into two parts. (**Divide**)
2. Then sort these and combine them. (**Conquer**)



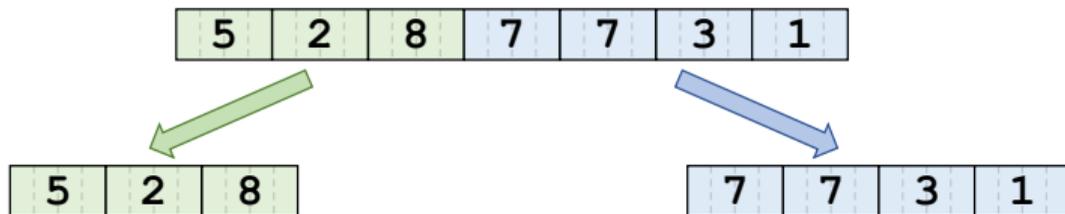
Merge-Sort

- Divide:

5	2	8	7	7	3	1
---	---	---	---	---	---	---

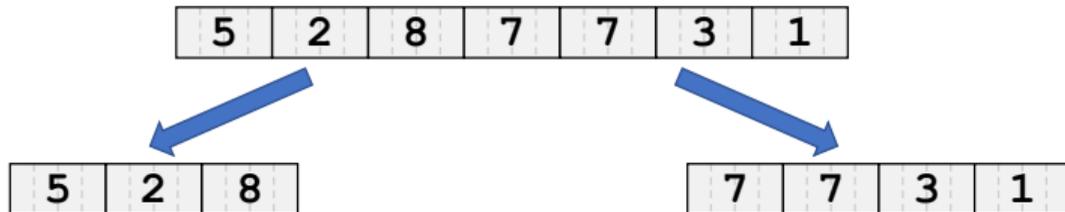
Merge-Sort

- Divide:



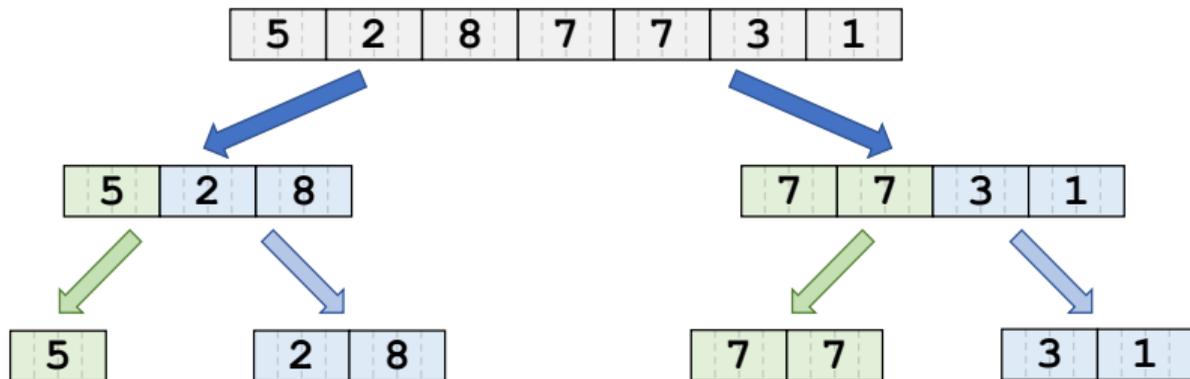
Merge-Sort

- Divide:



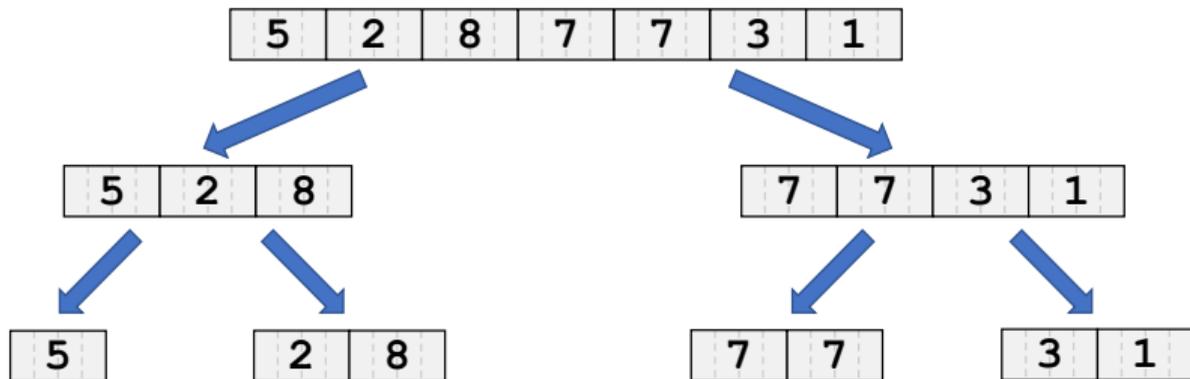
Merge-Sort

- Divide:



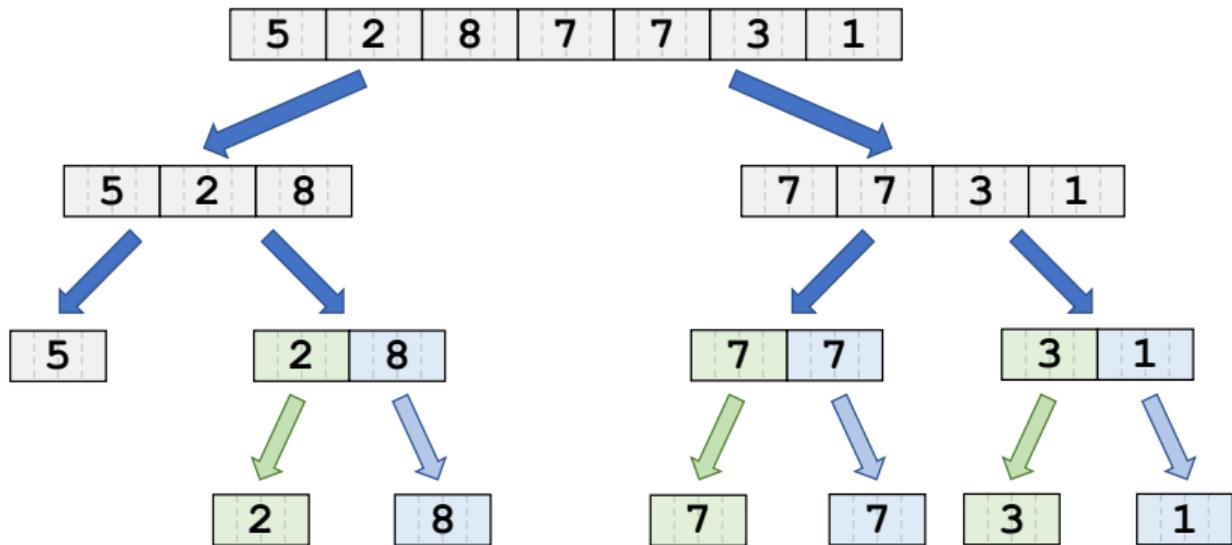
Merge-Sort

- Divide:



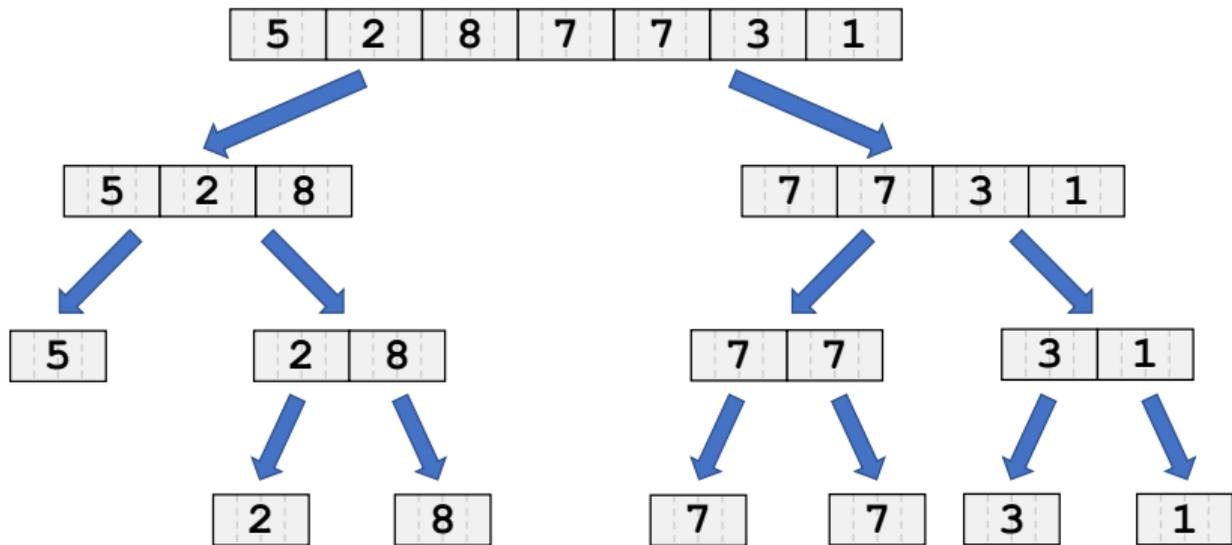
Merge-Sort

- Divide:



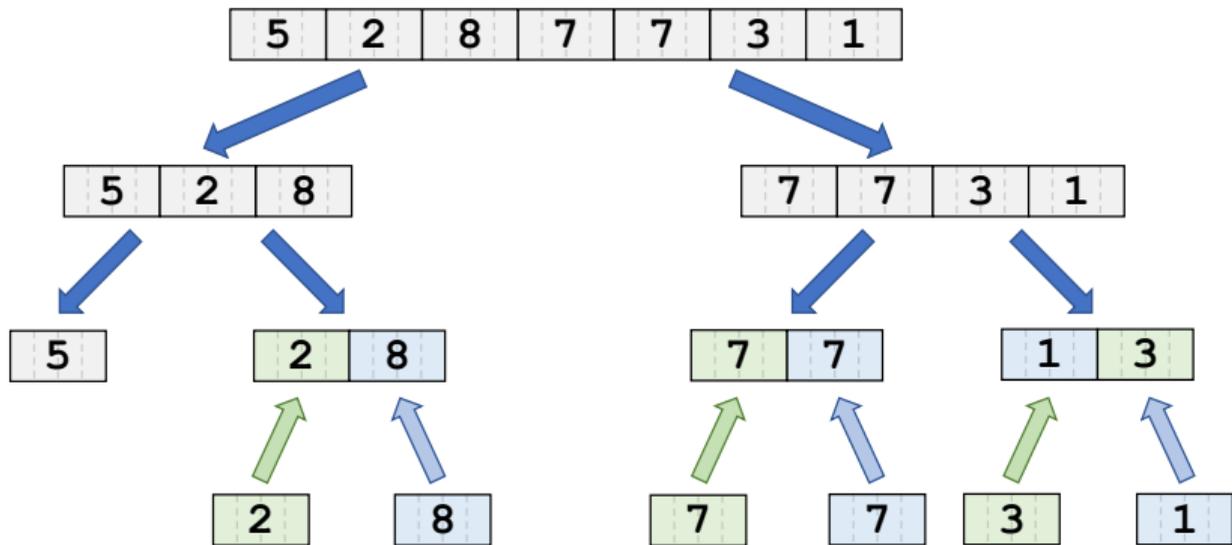
Merge-Sort

- Divide:



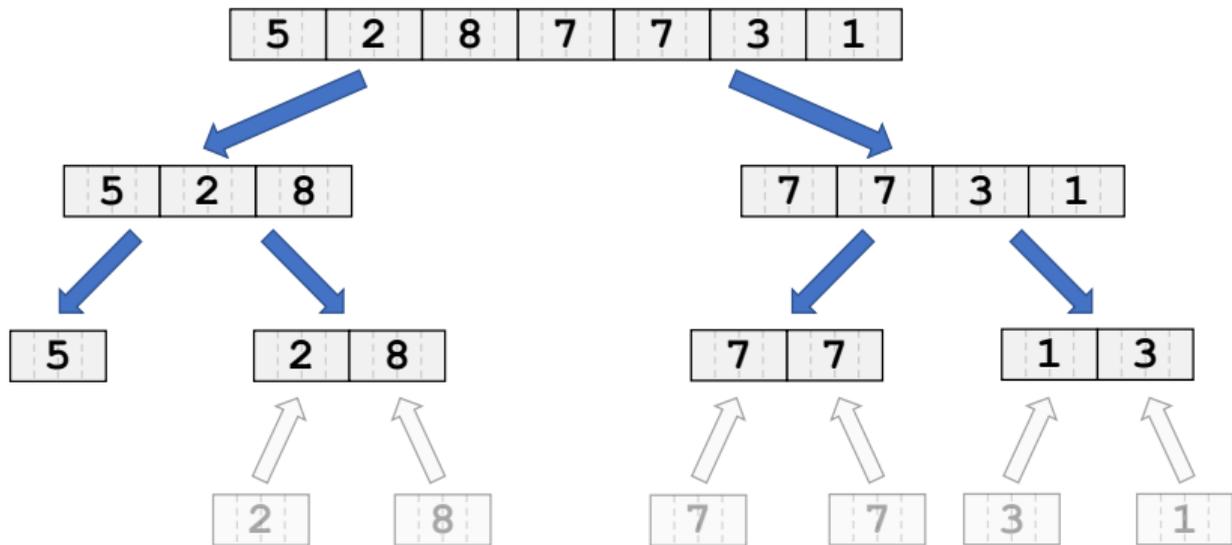
Merge-Sort

- Conquer:



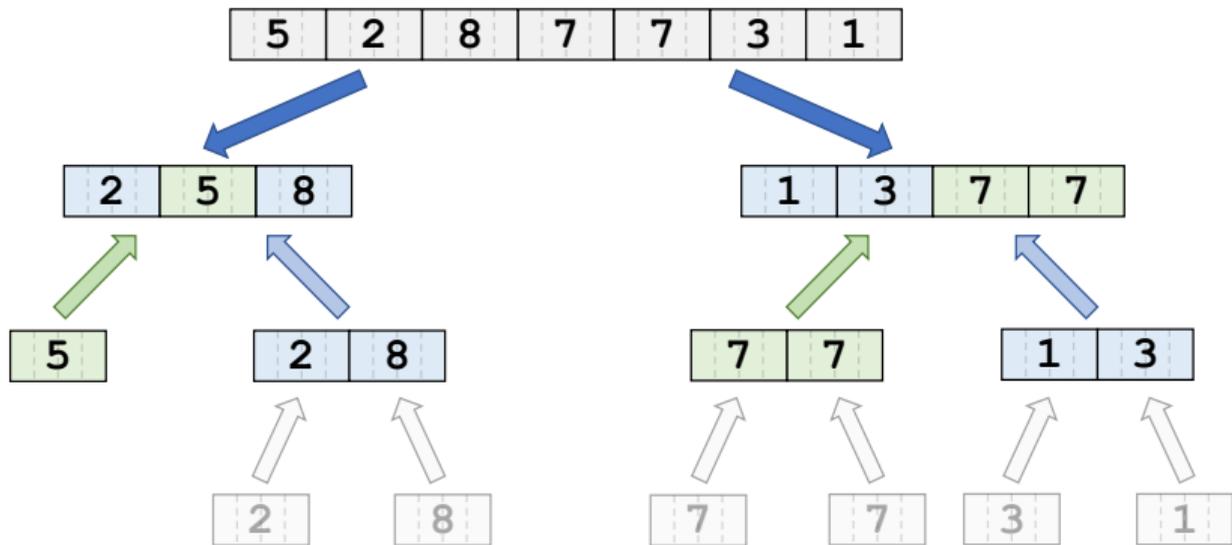
Merge-Sort

- Conquer:



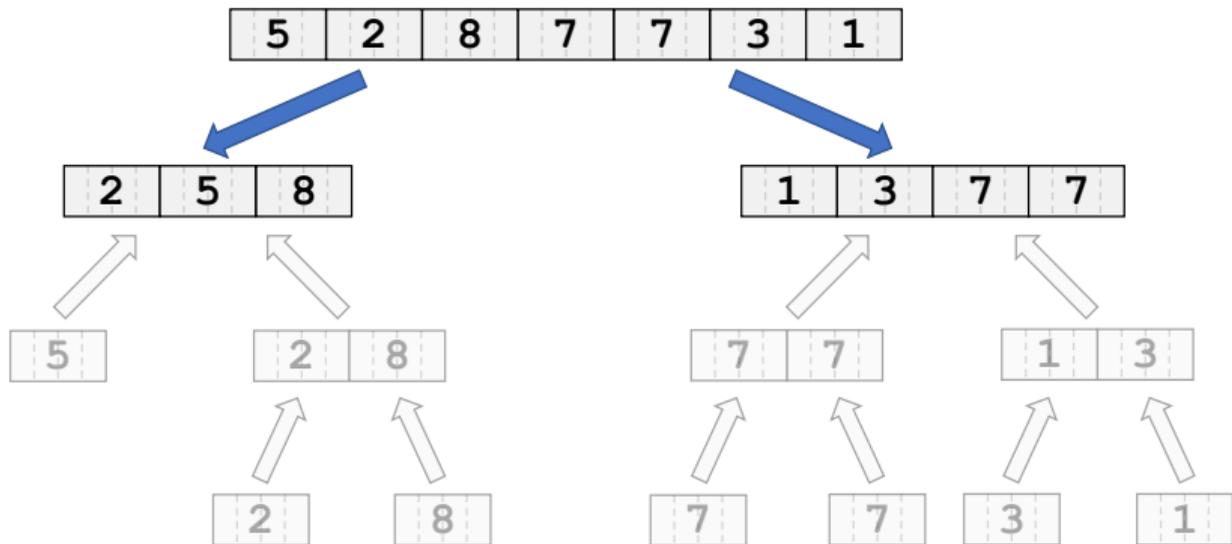
Merge-Sort

- Conquer:



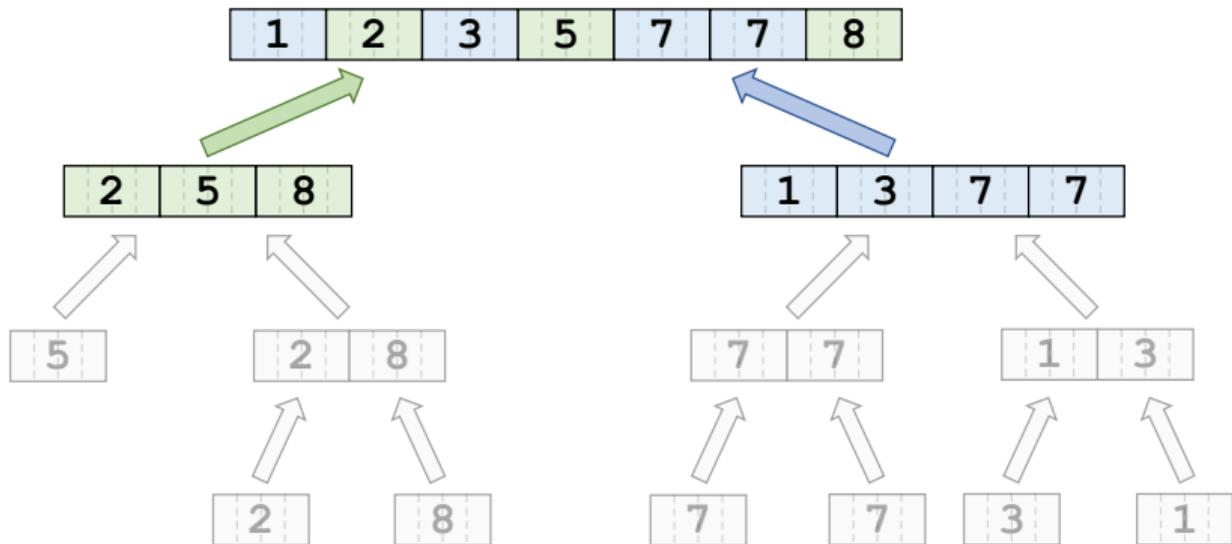
Merge-Sort

- Conquer:



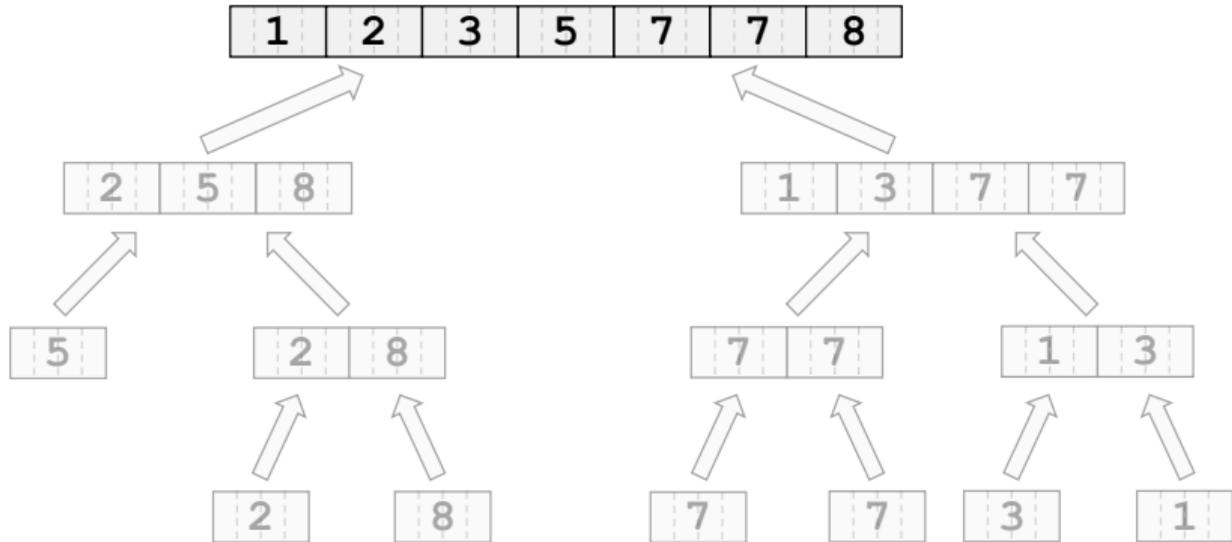
Merge-Sort

- Conquer:



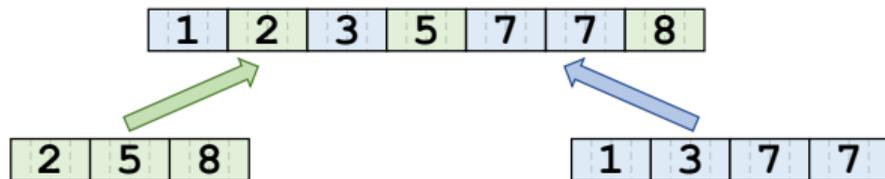
Merge-Sort

- Conquer:



Merge-Step

- How does



work?

- Card-player's trick:
Remove smaller «top card» (see next slides)

Merge-Sort – Exercise 1

- Idea:



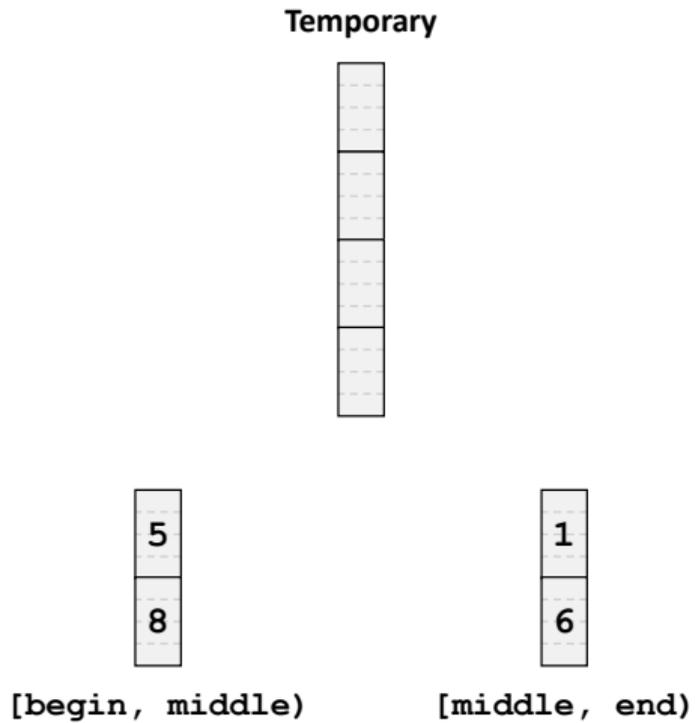
[begin, middle)



[middle, end)

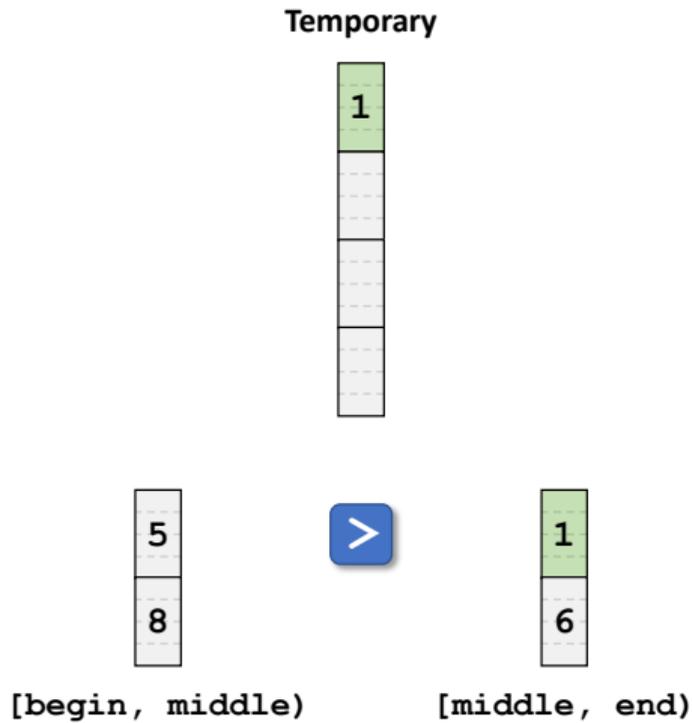
Merge-Sort – Exercise 1

- Idea:



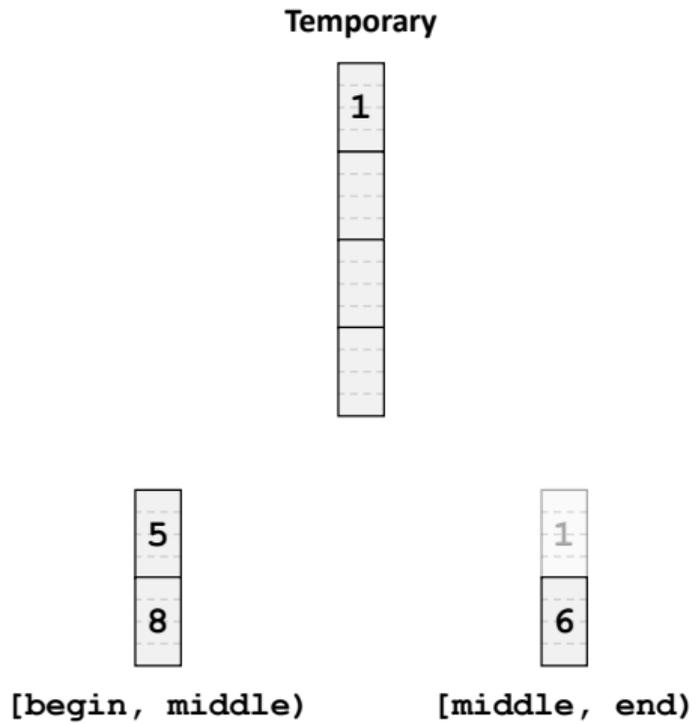
Merge-Sort – Exercise 1

- Idea:



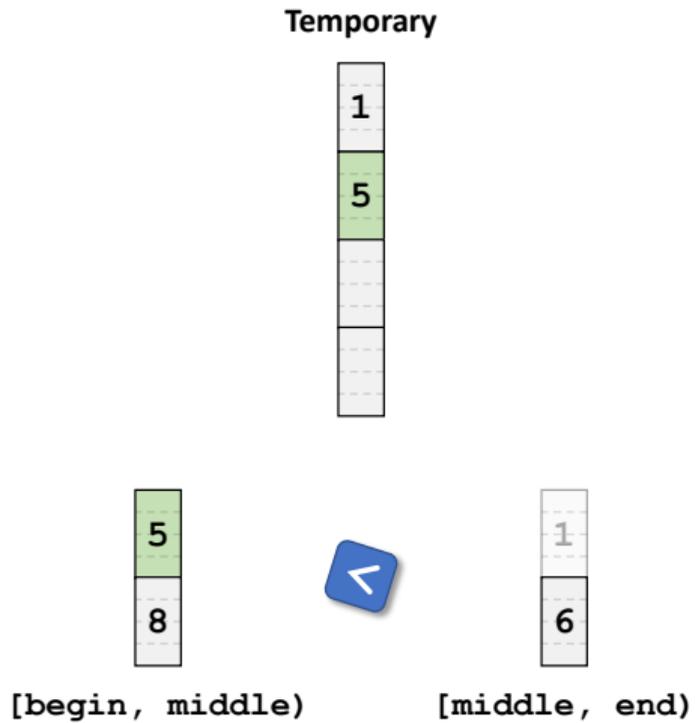
Merge-Sort – Exercise 1

- Idea:



Merge-Sort – Exercise 1

- Idea:



Merge-Sort – Exercise 1

- Idea:

Temporary



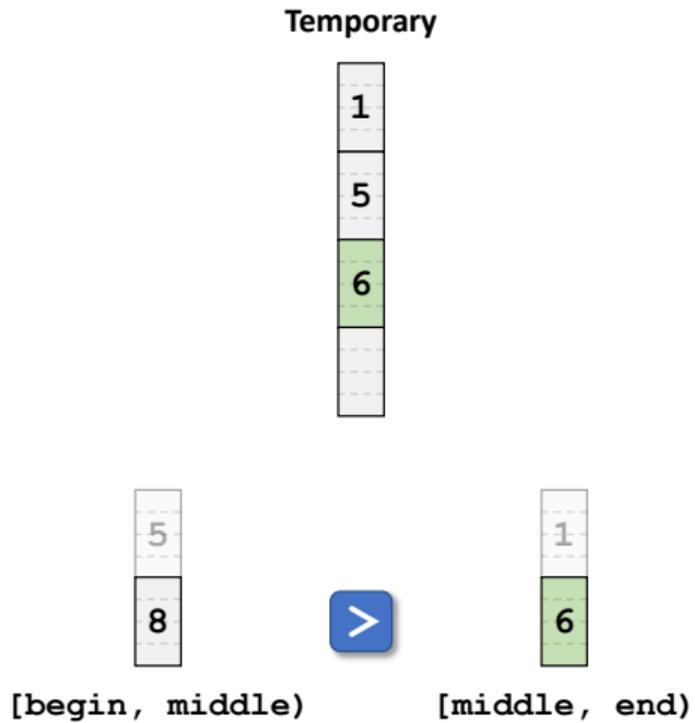
[begin, middle)



[middle, end)

Merge-Sort – Exercise 1

- Idea:



Merge-Sort – Exercise 1

- Idea:

Temporary



[begin, middle)



[middle, end)

Merge-Sort – Exercise 1

- Idea:

Temporary



[begin, middle)



[middle, end)

Merge-Sort – Exercise 1

- Idea:

Temporary



[begin, middle)

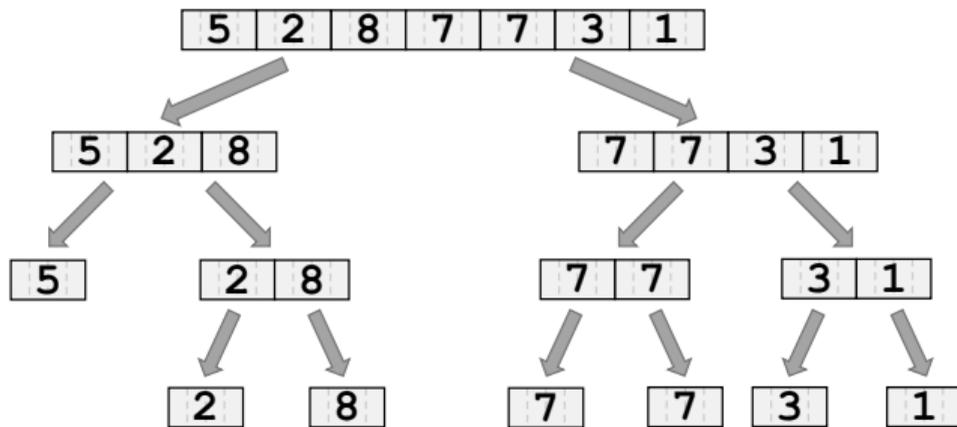


[middle, end)

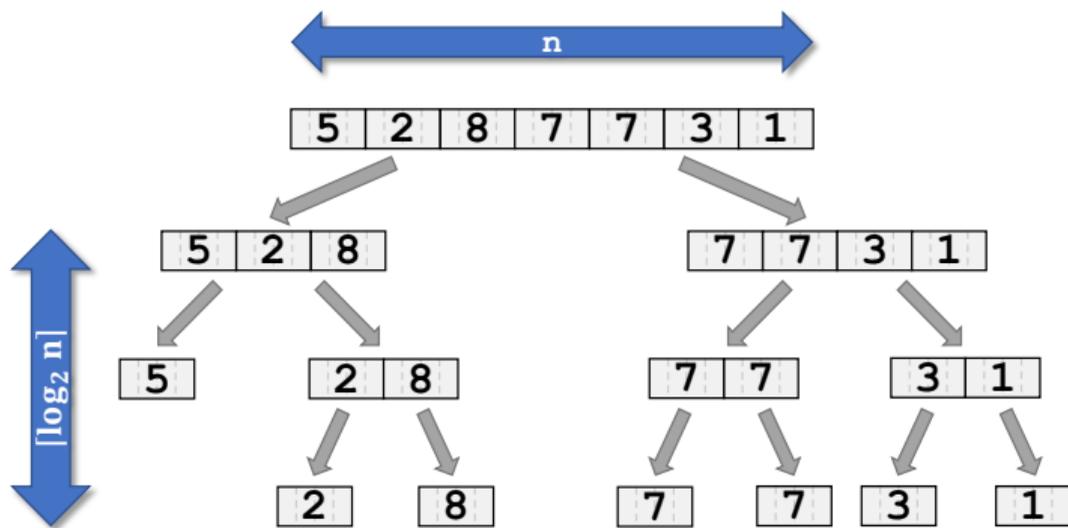
Runtime

(Intuition)

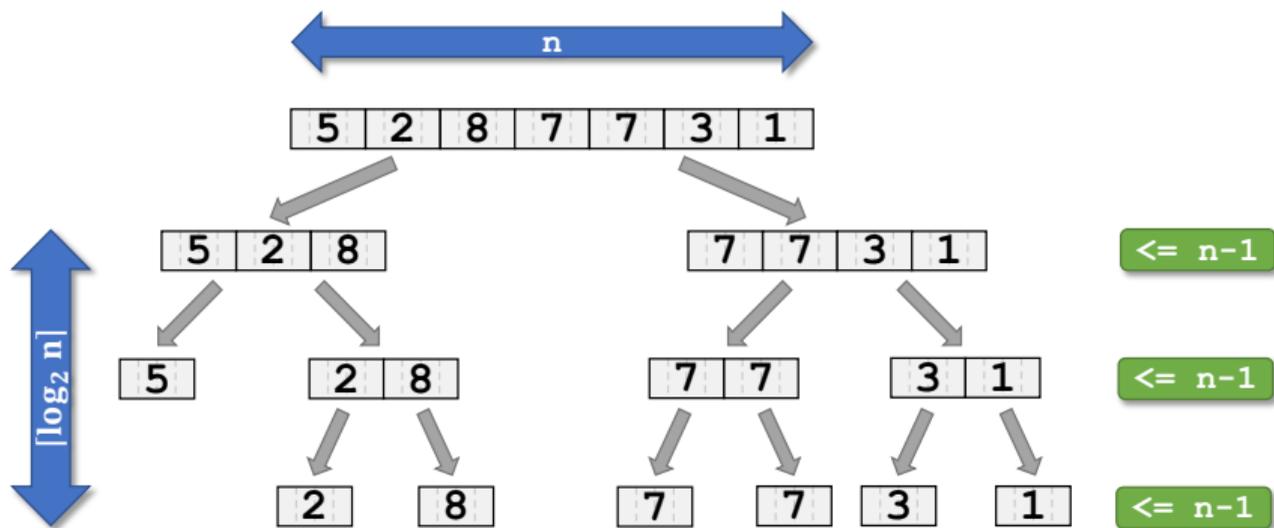
Alternative Proof



Alternative Proof

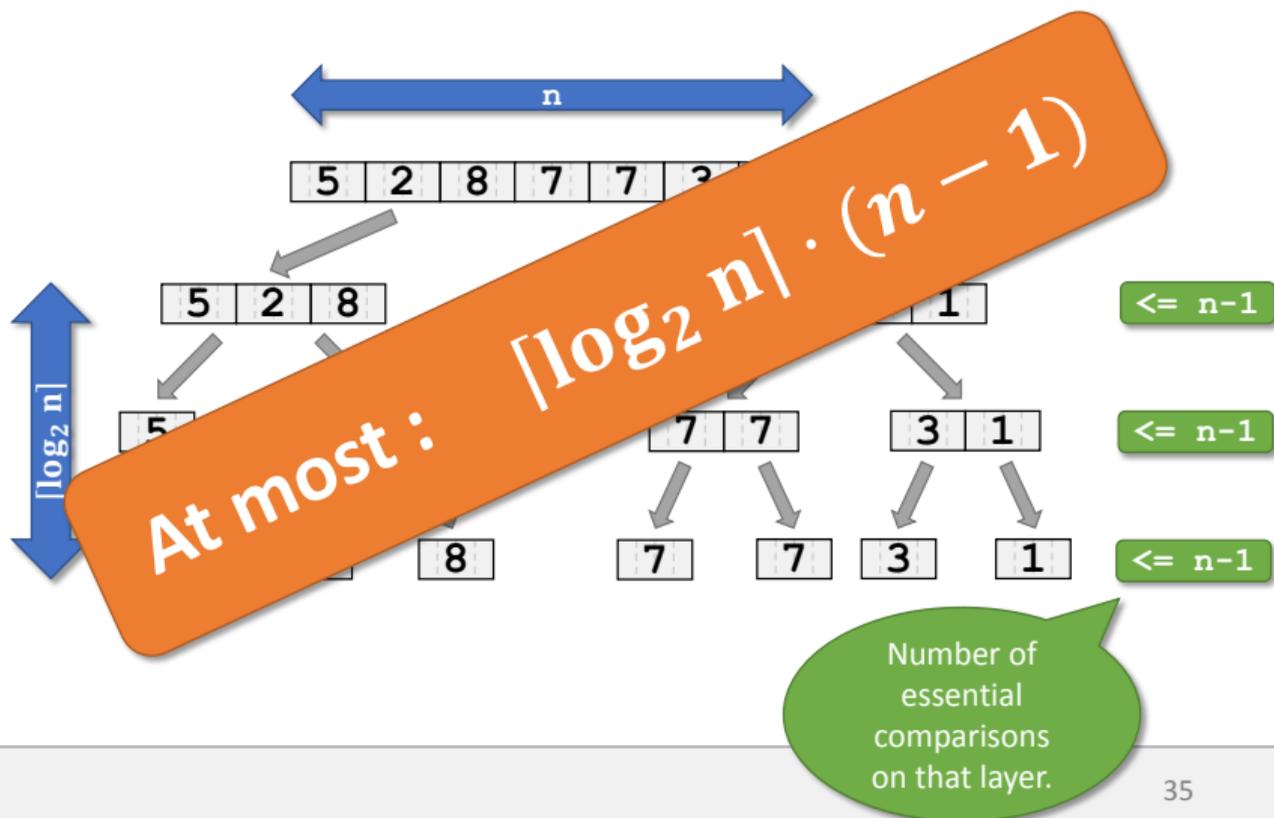


Alternative Proof



Number of essential comparisons on that layer.

Alternative Proof



Aufgabe "our_list::merge_sorted" (Schwierig)

Aufgabe "our_list::merge_sorted" (Schwierig)

- Öffnet "our_list::merge_sorted" auf **code expert**

Aufgabe "our_list::merge_sorted" (Schwierig)

- Öffnet "our_list::merge_sorted" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet

Aufgabe "our_list::merge_sorted" (Schwierig)

- Öffnet "our_list::merge_sorted" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet
- Programmiert eine Lösung (optional in Gruppen)

Aufgabe "our_list::merge_sorted" (Lösung)

Aufgabe "our_list::merge_sorted" (Lösung)

Siehe **code expert**

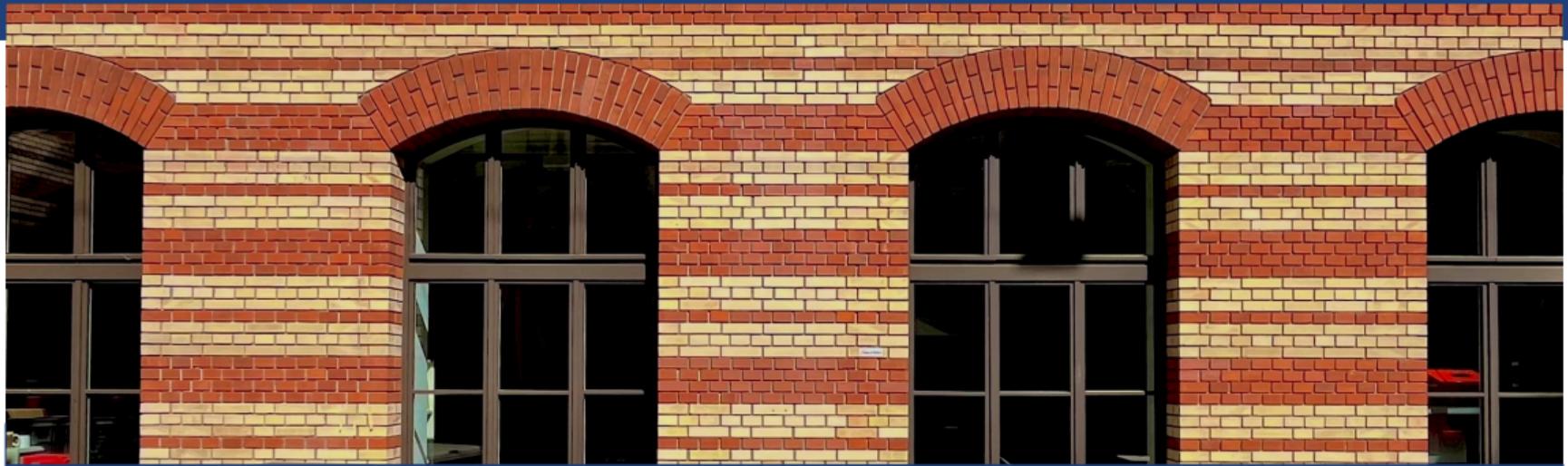
Fragen/Unklarheiten?

9. Outro

Allgemeine Fragen?

Bis zum nächsten Mal

Schöne Woche noch!



Übungsstunde — Informatik — 11

Adel Gavranović

Memory Management, Probleme mit Pointer, Smart Pointer, Muddiest Point

Übersicht

Follow-up
Memory Management
Aufgabe "Box"
Häufige Probleme mit Pointer
Shared and Unique Pointers
Muddiest Point



n.ethz.ch/~agavranovic

[Material](#)

[Webpage](#)

[Mail](#)

1. Follow-up

Fragen/Unklarheiten?

2. Feedback zu **code** expert

3. Lernziele

Ziele

- Programme die **new**, **delete**, copy constructors und destructors verwenden tracen können
- Dynamischen Speicher verstehen
- Die Ursachen und Gefahren von dangling pointers, double-free und use-after-free kennen und verhindern können
- `shared_ptr` verwenden können

4. Zusammenfassung

5. Memory Management

new und delete

Vergesst nie...

Zu jedem **new** ein **delete**

Vergesst nie...

Zu jedem **new** ein **delete**

Constructor, Copy-Constructor, Destructor

Vergesst nie...

Zu jedem `new` ein `delete`

Constructor, Copy-Constructor, Destructor

- Sind einfach Funktionen, die zu bestimmten Anlässen gecalled werden

Vergesst nie...

Zu jedem `new` ein `delete`

Constructor, Copy-Constructor, Destructor

- Sind einfach Funktionen, die zu bestimmten Anlässen gecalled werden
- Müssen `public` sein

Constructor

Constructor

- Wird gerufen, wenn

Constructor

Constructor

- Wird gerufen, wenn ein Objekt einer class/struct constructed wird

Constructor

- Wird gerufen, wenn ein Objekt einer class/struct constructed wird
- Man kann dem Constructor Argumente geben, um das Objekt genau so zu initialisieren wie wir wollen

Constructor

- Wird gerufen, wenn ein Objekt einer class/struct constructed wird
- Man kann dem Constructor Argumente geben, um das Objekt genau so zu initialisieren wie wir wollen
- Man kann mehrere Constructoren haben, z.B. für verschiedene Typen. Der Computer schliesst dann auf den richtigen Typ. Beispiel:
 - `personClass Person001(142.0f);`
 - `personClass Person161(420);`

Constructor

Constructor

- Wird gerufen, wenn ein Objekt einer class/struct constructed wird
- Man kann dem Constructor Argumente geben, um das Objekt genau so zu initialisieren wie wir wollen
- Man kann mehrere Constructoren haben, z.B. für verschiedene Typen. Der Computer schliesst dann auf den richtigen Typ. Beispiel:
 - `personClass Person001(142.0f);`
 - `personClass Person161(420);`
- Mehr dazu: [cppreference link](#)

Constructor - Beispiel in einer class

Constructor - Beispiel in einer class

```
class meineKlasse {
    int a, b;
public:
    const int& r; // for reading only!

    // CONSTRUCTOR
    meineKlasse(int i)
        : a(i)      // initializes a to the value of i
        , b(i+5)    // initializes b to the value of i+5
        , r(a)      // initializes r to refer to a
        // ^ here we are using a "member initializer list"
        // and if you want your constructor to do
        // anything additionally, put it inside
        { /*here (like in a regular function!)*/* }
};
```

Member Initializer List

```
meineKlasse::meineKlasse()  
    : memberVariableEins(0)           // init memberVariableEins  
    { memberVariableZwei = 0; }      // init memberVariableZwei
```

Was ist der Unterschied zwischen diesen beiden Initialisierungen der Membervariablen?

Member Initializer List

```
meineKlasse::meineKlasse()  
    : memberVariableEins(0)           // init memberVariableEins  
    { memberVariableZwei = 0; }      // init memberVariableZwei
```

Was ist der Unterschied zwischen diesen beiden Initialisierungen der Membervariablen? Wieso machen wir uns die Mühe, MILs zu verwenden?

Member Initializer List

```
meineKlasse::meineKlasse()  
    : memberVariableEins(0)           // init memberVariableEins  
    { memberVariableZwei = 0; }      // init memberVariableZwei
```

Was ist der Unterschied zwischen diesen beiden Initialisierungen der Membervariablen? Wieso machen wir uns die Mühe, MILs zu verwenden?

const members

- In manchen Fällen möchte man **const** Member haben und die zweite Option würde dort nicht funktionieren

Member Initializer List

```
meineKlasse::meineKlasse()  
    : memberVariableEins(0)           // init memberVariableEins  
    { memberVariableZwei = 0; }      // init memberVariableZwei
```

Was ist der Unterschied zwischen diesen beiden Initialisierungen der Membervariablen? Wieso machen wir uns die Mühe, MILs zu verwenden?

const members

- In manchen Fällen möchte man **const** Member haben und die zweite Option würde dort nicht funktionieren

Performance

- Der Hauptgrund für uns ist Performance. Der Code mit MILs ist schneller, da er unnötige Kopien vermeidet. Diese Kopien sehen wir nicht im Code, aber die Laufzeit/Performance wird dadurch schlechter

Destructor

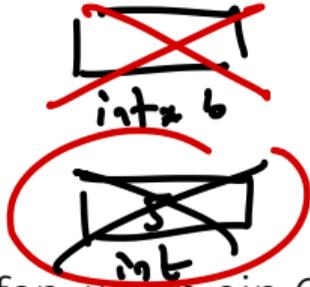
- wird gerufen, wenn

Destructor

- wird gerufen, wenn ein Objekt einer class/struct *destructured* wird. Das kann passieren,

Destructor

Destructor



```
}  
int a = 5;  
→ int* b = new int(5);  
→ delete b;
```

- wird gerufen, wenn ein Objekt einer class/struct destructured wird. Das kann passieren, am Ende eines Scopes oder wenn **delete** verwendet wird

```
→ { P obj(5); // "normal"  
→ P* ptr = new P(5); // "dynamic"
```

```
delete ptr;  
} ←
```

Destructor

- wird gerufen, wenn ein Objekt einer class/struct *destructured* wird. Das kann passieren, am Ende eines Scopes oder wenn **delete** verwendet wird
- Wird genutzt, um Memory "sauber" zu halten, wenn ein Objekt nicht länger verwendet wird

Destructor - Beispiel in einer class

Destructor - Beispiel in einer class

```
class meineKlasse {  
    int* value;  
  
public:  
  
    // other -ctors and stuff go here  
  
→ ~meineKlasse(){  
    *value = 7;  
    delete value; // That's how we clean up the value which  
                  // lies at the slot that the int-pointer is  
                  // pointing to, instead of just deleting  
                  // the int-pointer (avoiding "memory leaks")  
}  
};
```

int value*

5
int

delete **value;*

delete erwartet eine Adresse (pointe!!!)

Copy-constructor

Copy-Constructor

- wird gerufen, wenn

Copy-constructor

Copy-Constructor

- wird gerufen, wenn ein Objekt mit einem anderen Objekt derselben class/struct *initialisiert* wird

Copy-constructor

Copy-Constructor

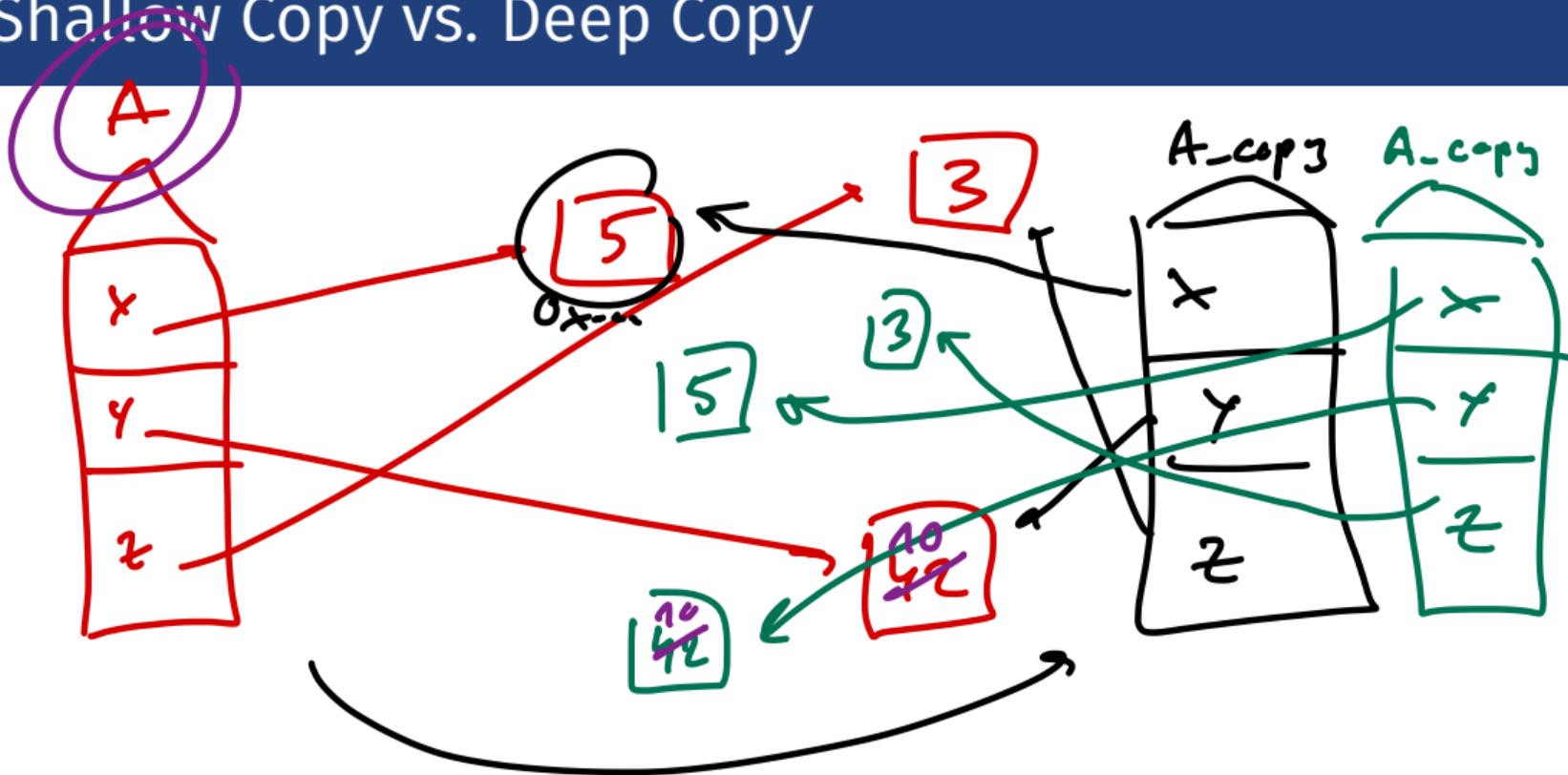
- wird gerufen, wenn ein Objekt mit einem anderen Objekt derselben class/struct *initialisiert* wird
- es gibt ein default copy-constructor, falls wir keinen explizit deklarieren. Dieser macht einfach eine member-wise copy der class/struct
- lässt uns präzise bestimmen, wie wir etwas kopieren möchten statt einfach eine *shallow copy* zu machen

Copy-constructor

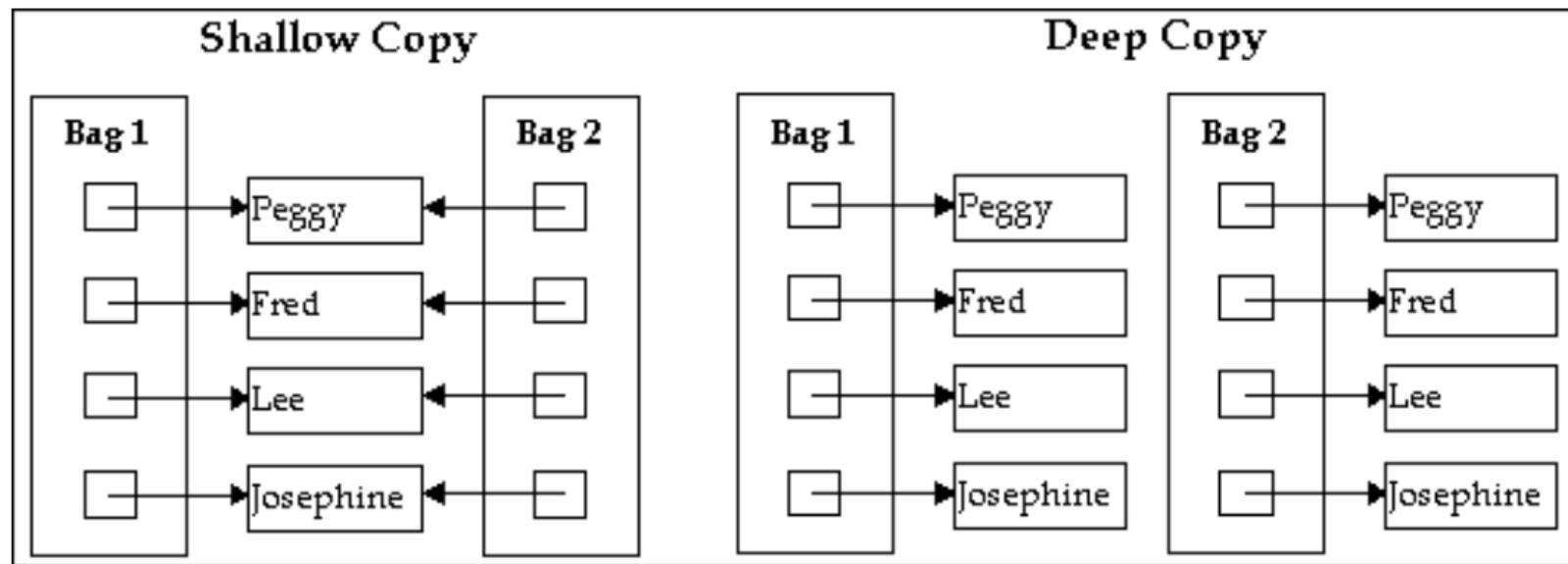
Copy-Constructor

- wird gerufen, wenn ein Objekt mit einem anderen Objekt derselben class/struct *initialisiert* wird
- es gibt ein default copy-constructor, *falls* wir keinen explizit deklarieren. Dieser macht einfach eine member-wise copy der class/struct
- lässt uns präzise bestimmen, wie wir etwas kopieren möchten statt einfach eine *shallow copy* zu machen
- nicht zu verwechseln mit dem **operator=**, der etwas sehr ähnliches macht

Shallow Copy vs. Deep Copy



Shallow Copy vs. Deep Copy



(copy-)assignment-operator (=)

Assignment-operator (=)

- wird gerufen, wenn

(copy-)assignment-operator (=)

Assignment-operator (=)

- wird gerufen, wenn ein Objekt einem anderen Objekt derselben class/struct *assigned* wird

(copy-)assignment-operator (=)

Assignment-operator (=)

- wird gerufen, wenn ein Objekt einem anderen Objekt derselben class/struct *assigned* wird
- wird *nur nach* (nicht bei) Initialisierungen gerufen

(copy-)assignment-operator (=)

Assignment-operator (=)

- wird gerufen, wenn ein Objekt einem anderen Objekt derselben class/struct *assigned* wird
- wird *nur nach* (nicht bei) Initialisierungen gerufen
- heisst "assignment operator", so wie bei primitiven Types (=)

(copy-)assignment-operator (=)

Assignment-operator (=)

- wird gerufen, wenn ein Objekt einem anderen Objekt derselben class/struct *assigned* wird
- wird *nur nach* (nicht bei) Initialisierungen gerufen
- heisst "assignment operator", so wie bei primitiven Types (=)
- Faustregel: macht erst destructor-Zeugs, dann copy-constructor-Zeugs

(copy-)assignment-operator (=)

Assignment-operator (=)

- wird gerufen, wenn ein Objekt einem anderen Objekt derselben class/struct *assigned* wird
- wird *nur nach* (nicht bei) Initialisierungen gerufen
- heisst "assignment operator", so wie bei primitiven Types (=)
- Faustregel: macht erst destructor-Zeugs, dann copy-constructor-Zeugs
- *muss* einen return type haben, meist

(copy-)assignment-operator (=)

Assignment-operator (=)

- wird gerufen, wenn ein Objekt einem anderen Objekt derselben class/struct *assigned* wird
- wird *nur nach* (nicht bei) Initialisierungen gerufen
- heisst "assignment operator", so wie bei primitiven Types (=)
- Faustregel: macht erst destructor-Zeugs, dann copy-constructor-Zeugs
- *muss* einen return type haben, meist `class&` damit

(copy-)assignment-operator (=)

Assignment-operator (=)

- wird gerufen, wenn ein Objekt einem anderen Objekt derselben class/struct *assigned* wird
- wird *nur nach* (nicht bei) Initialisierungen gerufen
- heisst "assignment operator", so wie bei primitiven Types (=)
- Faustregel: macht erst destructor-Zeugs, dann copy-constructor-Zeugs
- *muss* einen return type haben, meist **class&** damit man *chained assignments* machen kann (`a = b = (c = d);`, allen wird d assigned)

`a = (b = (c = d));`
`a => b = c = d`

operator= vs. Copy-Constructor

```
// our class/struct is named "Box"

Box first;           // init by default constructor
Box second(first);  // init by copy-constructor
Box third = first;  // also init by copy-constructor
second = third;     // assignment by (copy-)assignment operator
```

operator= vs. Copy-Constructor

Info: slides are online now!

```
// our class/struct is named "Box"

Box first;           // init by default constructor
Box second(first);  // init by copy-constructor
Box third = first;  // also init by copy-constructor
second = third;     // assignment by (copy-)assignment operator
```

Die letzten beiden Fälle sehen ähnlich aus, aber denkt daran:
dass der (copy-)assignment-operator= erst in Aktion tritt, *nachdem* ein
Objekt bereits initialisiert wurde

Fragen/Unklarheiten?

6. Aufgabe "Box"

Aufgabe "Box (copy)"

Hier schauen wir uns die Implementation *sehr* genau an.

- Öffnet **code expert** und das Code-Beispiel "Box (copy)"

Aufgabe "Box (copy)"

Hier schauen wir uns die Implementation *sehr* genau an.

- Öffnet **code expert** und das Code-Beispiel "Box (copy)"
- Macht euch noch keine Gedanken über `main.cpp`, dazu kommen wir noch

Aufgabe "Box (copy)"

Hier schauen wir uns die Implementation *sehr* genau an.

- Öffnet **code expert** und das Code-Beispiel "Box (copy)"
- Macht euch noch keine Gedanken über `main.cpp`, dazu kommen wir noch
- Macht euch auch keine Gedanken über `std::cerr`, es ist nur ein fancy `std::cout`

Aufgabe "Box (copy)"

Hier schauen wir uns die Implementation *sehr* genau an.

- Öffnet **code expert** und das Code-Beispiel "Box (copy)"
- Macht euch noch keine Gedanken über `main.cpp`, dazu kommen wir noch
- Macht euch auch keine Gedanken über `std::cerr`, es ist nur ein fancy `std::cout`
- Kleines code-together :)

Members von "Box"

```
Box::Box(const Box& other) {  
    ptr = new int(*other.ptr);  
}  
  
Box& Box::operator= (const Box& other) {  
    *ptr = *other.ptr;  
    return *this;  
}
```

Members von "Box"

```
Box::~~Box() {  
    delete ptr;  
    ptr = nullptr;  
}  
  
Box::Box(int* v) {  
    ptr = v;  
}  
  
int& Box::value() {  
    return *ptr;  
}
```

Tracing test_destructor1()

```
void test_destructor1() {
    std::cerr << "[enter] test_destructor1" << std::endl;

    int a;

    {
        Box box(new int(1));
        a = 5;
    }

    std::cout << "a = " << a << std::endl;
    std::cerr << "[exit] test_destructor1" << std::endl;
}
```

Tracing test_destructor2()

```
void test_destructor2() {
    std::cerr << "[enter] test_destructor2" << std::endl;

    {
        Box* box_ptr = new Box(new int(2));
        delete box_ptr;    // to trigger destructor of Box above
    }

    std::cerr << "[exit] test_destructor2" << std::endl;
}
```

Tracing test_copy_constructor()

```
void test_copy_constructor() {
    std::cerr << "[enter] test_copy_constructor" << std::endl;

    {
        Box demo(new int(0));
        Box demo_copy = demo;

        demo.value() = 4;

        demo_copy.value() = 5;
    }

    std::cerr << "[exit] test_copy_constructor" << std::endl;
}
```

Tracing test_assignment()

```
void test_assignment() {
    std::cerr << "[enter] test_assignment" << std::endl;

    {
        Box demo(new int(0));
        demo.value() = 3;
        Box demo_copy(new int(0));
        demo_copy = demo;
        demo.value() = 4;
        demo_copy.value() = 5;
    }

    std::cerr << "[exit] test_assignment" << std::endl;
}
```

Fragen/Unklarheiten?

7. Häufige Probleme mit Pointer

Dangling Pointers

Was?

¹Oft ein *Zombie* genannt

Dangling Pointers

Was?

Ein *Dangling Pointer* ist ein Pointer der auf eine Speicherstelle zeigt, die bereits dealloziert wurde, also zeigt er auf eine invalide Speicherstelle.¹

Wie?

¹Oft ein *Zombie* genannt

Dangling Pointers

Was?

Ein *Dangling Pointer* ist ein Pointer der auf eine Speicherstelle zeigt, die bereits dealloziert wurde, also zeigt er auf eine invalide Speicherstelle.¹

Wie?

Dies tritt häufig auf, wenn ein Objekt gelöscht wird oder aus dem Scope verschwindet, aber der Zeiger, der darauf zeigt, nicht auf `nullptr` gesetzt wird. So zeigt der Pointer immer noch auf den alten Speicherplatz, obwohl er nicht weiss, was sich jetzt dort befindet.

Na und?

¹Oft ein *Zombie* genannt

Dangling Pointers

Was?

Ein *Dangling Pointer* ist ein Pointer der auf eine Speicherstelle zeigt, die bereits dealloziert wurde, also zeigt er auf eine invalide Speicherstelle.¹

Wie?

Dies tritt häufig auf, wenn ein Objekt gelöscht wird oder aus dem Scope verschwindet, aber der Zeiger, der darauf zeigt, nicht auf `nullptr` gesetzt wird. So zeigt der Pointer immer noch auf den alten Speicherplatz, obwohl er nicht weiss, was sich jetzt dort befindet.

Na und?

Der Zugriff oder die Manipulation eines *Dangling Pointer* kann zu unvorhersehbarem Verhalten, Abstürzen oder Datenbeschädigung führen, da der Speicher möglicherweise neu zugewiesen wurde.

¹Oft ein *Zombie* genannt

Double-Free

Was?

Double-Free

Was?

Double-free tritt auf, wenn `delete` zweimal für dieselbe Speicherzuweisung aufgerufen wird.

Wie?

Double-Free

Was?

Double-free tritt auf, wenn `delete` zweimal für dieselbe Speicherzuweisung aufgerufen wird.

Wie?

Dies tritt häufig in komplexen Programmen auf, bei denen die Speicherverwaltung an mehreren Stellen erfolgt, was zu Verwirrung darüber führt, wer über den Speicherplatz verfügt.

Na und?

Double-Free

Was?

Double-free tritt auf, wenn **delete** zweimal für dieselbe Speicherzuweisung aufgerufen wird.

Wie?

Dies tritt häufig in komplexen Programmen auf, bei denen die Speicherverwaltung an mehreren Stellen erfolgt, was zu Verwirrung darüber führt, wer über den Speicherplatz verfügt.

Na und?

Das doppelte Freigeben von Speicher kann die Speicherzuweisungsmetadaten beschädigen, was zu Speicherlecks, Programmabstürzen oder anderem fehlerhaften Verhalten führen kann.

Use-After-Free

Was?

Use-After-Free

Was?

Use-after-free ist eine Situation, in der ein Programm einen Zeiger weiter verwendet, nachdem es den Speicher, auf den er zeigt, freigegeben hat.

Wie?

Use-After-Free

Was?

Use-after-free ist eine Situation, in der ein Programm einen Zeiger weiter verwendet, nachdem es den Speicher, auf den er zeigt, freigegeben hat.

Wie?

Dies kann passieren, wenn das Programm den Zeiger nicht auf `nullptr` setzt, nachdem es ihn freigegeben hat, oder wenn es Kopien des Zeigers gibt, die nicht aktualisiert wurden.

Na und?

Use-After-Free

Was?

Use-after-free ist eine Situation, in der ein Programm einen Zeiger weiter verwendet, nachdem es den Speicher, auf den er zeigt, freigegeben hat.

Wie?

Dies kann passieren, wenn das Programm den Zeiger nicht auf `nullptr` setzt, nachdem es ihn freigegeben hat, oder wenn es Kopien des Zeigers gibt, die nicht aktualisiert wurden.

Na und?

Da der freigegebene Speicher möglicherweise für andere Zwecke neu zugewiesen wurde, kann seine Verwendung zu Datenbeschädigung, unvorhersehbarem Programmverhalten oder Sicherheitslücken führen.

*nullptr

*nullptr



 xkcd

Fragen/Unklarheiten?

Dazu verdammt, Fehler zu machen?

Dazu verdammt, Fehler zu machen?

Wie kann man das alles verhindern?

Dazu verdammt, Fehler zu machen?

Wie kann man das alles verhindern?

Smart Pointers!

8. Shared and Unique Pointers

Smart Pointers

- Smart pointers sind bequeme Wrapper um reguläre Pointer, die helfen, Speicherlecks zu verhindern, indem sie den Speicher automatisch verwalten.
- Die Smart-Pointer `shared_ptr` und `weak_ptr` sind Teil der Standardbibliothek `<memory>`

Vergleich `unique_ptr` VS `shared_ptr`

`shared_ptr`

Vergleich `unique_ptr` VS `shared_ptr`

`shared_ptr`

Ein `shared_ptr` erlaubt es mehreren Zeigern, sich das Eigentum an derselben Ressource zu teilen. Es wird gezählt, wie viele Zeiger auf dieselbe Ressource zeigen. Sobald die Anzahl 0 erreicht, wird das Objekt gelöscht.

Vergleich `unique_ptr` VS `shared_ptr`

`shared_ptr`

Ein `shared_ptr` erlaubt es mehreren Zeigern, sich das Eigentum an derselben Ressource zu teilen. Es wird gezählt, wie viele Zeiger auf dieselbe Ressource zeigen. Sobald die Anzahl 0 erreicht, wird das Objekt gelöscht.

```
std::shared_ptr<SomeClass> s1 = std::make_shared<SomeClass>()
```

Vergleich `unique_ptr` VS `shared_ptr`

`shared_ptr`

Ein `shared_ptr` erlaubt es mehreren Zeigern, sich das Eigentum an derselben Ressource zu teilen. Es wird gezählt, wie viele Zeiger auf dieselbe Ressource zeigen. Sobald die Anzahl 0 erreicht, wird das Objekt gelöscht.

```
std::shared_ptr<SomeClass> s1 = std::make_shared<SomeClass>()
```

`unique_ptr`

Vergleich `unique_ptr` VS `shared_ptr`

`shared_ptr`

Ein `shared_ptr` erlaubt es mehreren Zeigern, sich das Eigentum an derselben Ressource zu teilen. Es wird gezählt, wie viele Zeiger auf dieselbe Ressource zeigen. Sobald die Anzahl 0 erreicht, wird das Objekt gelöscht.

```
std::shared_ptr<SomeClass> s1 = std::make_shared<SomeClass>()
```

`unique_ptr`

Ein `unique_ptr` wird für exklusives Eigentum verwendet. Speicher, der mit einem `unique_ptr` verbunden ist, wird automatisch freigegeben, wenn er den Scope verlässt.

Fragen/Unklarheiten?

9. Muddiest Point

Woran hakt es bei dir?

Q&A Session

10. Outro

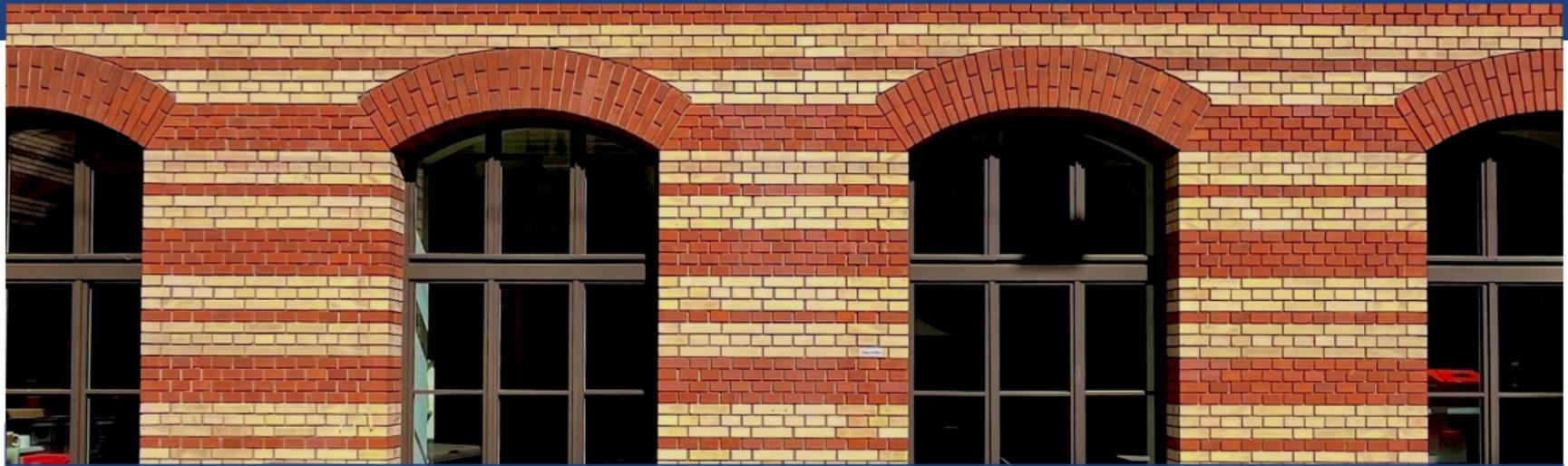
Allgemeine Fragen?

Bis zum nächsten Mal

Follow-up:

- Q&A (muddiest point)
- shared ptr
- Snack (Saviklaus?)

Schöne Woche noch!



Exercise Session — Computer Science — 12

Adel Gavranović

Pointer-Arithmetik, Speichermanagement

Übersicht

Follow-up

Pointers

Example: Pointers on Arrays

Example: Special Copy

Aufgabe "Push Back"

Memory Management

Muddiest Point



`n.ethz.ch/~agavranovic`

 Material

 Webpage

 Mail

1. Intro

- Any new people here today?

2. Follow-up

Muddiest Point

Muddiest Point

Schauen wir uns am Ende der Session nochmal kurz an

Shared Pointers

Shared Pointers

- Das Code-Beispiel "Shared Pointers (explanation)" illustriert die Verwendung und den Nutzen von Shared Pointern ziemlich gut
- Falls doch jemand fragen hat, könnt ihr sie jetzt stellen

Fragen/Unklarheiten?

3. Lernziele

Ziele

- Programme die `new []`, `delete []`, schreiben und tracen können
- Den Unterschied zwischen `new` und `new []`, und `delete` und `delete []` verstehen
- Programme die Pointer-Arithmetik verwenden schreiben und tracen können

4. Zusammenfassung

5. Pointers

new VS new []

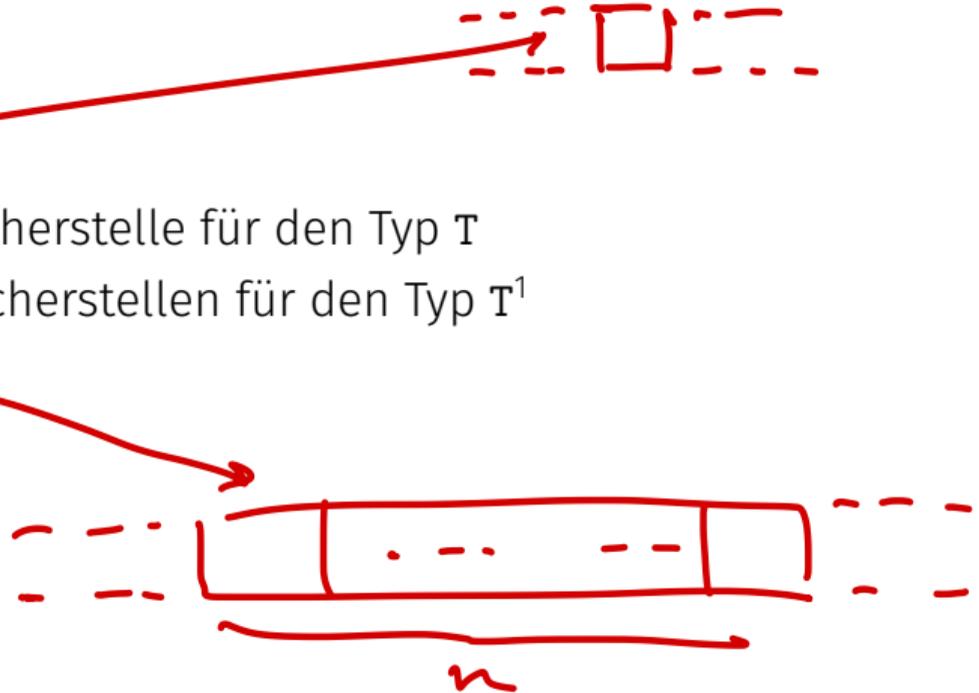
- `new` T alloziert **eine** Speicherstelle für den Typ T
- return pointer to newly alloc. object

¹diese Speicherstellen werden *zusammenhängend* sein, d. h. "nebeneinander" im Speicher

new VS new []

- `new T` alloziert **eine** Speicherstelle für den Typ T
- `new T[n]` alloziert **n** Speicherstellen für den Typ T¹

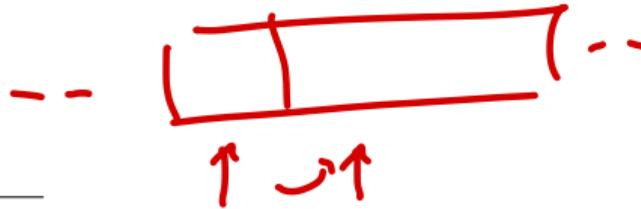
n
↳ return



¹diese Speicherstellen werden *zusammenhängend* sein, d. h. "nebeneinander" im Speicher

new VS new []

- `new T` alloziert **eine** Speicherstelle für den Typ `T`
- `new T[n]` alloziert **n** Speicherstellen für den Typ `T`¹ ←
- Beide geben einen Pointer zurück, bei einer Range zeigt dieser auf das erste Objekt



→ ¹diese Speicherstellen werden zusammenhängend sein, d. h. "nebeneinander" im Speicher

Arrays

Statisch allozierter Array

Statisch allozierter Array

```
int myStatArr[3] = {2, 3, 8};
```

- myStatArr zeigt nun auf die

Arrays

Statisch allozierter Array

```
int myStatArr[3] = {2, 3, 8};
```

- `myStatArr` zeigt nun auf die 2
- `*myStatArr` gibt

Arrays

Statisch allozierter Array

```
int myStatArr[3] = {02, 13, 28};
```

- ⚡ ■ myStatArr zeigt nun auf die 2
 - *myStatArr gibt 2
 - myStatArr[2] gibt

Arrays

Statisch allozierter Array

```
int myStatArr[3] = {2, 7, 8};
```

Handwritten annotations: -4 above the second element, 0, 1, 2 below the elements.

- `myStatArr` zeigt nun auf die 2
- `*myStatArr` gibt 2
- `myStatArr[2]` gibt 8
- `myStatArr[1]` = -4

Handwritten annotations: A bracket under `myStatArr[1]` points to the text "read & write". Below "read" and "write" are two upward-pointing arrows.

Statisch allozierter Array

```
int myStatArr[3] = {2, 3, 8};
```

- `myStatArr` zeigt nun auf die 2
- `*myStatArr` gibt 2
- `myStatArr[2]` gibt 8
- `myStatArr[1] = -4` setzt 3 auf -4

Statisch allozierter Array

```
int myStatArr[3] = {2, 3, 8};
```

- `myStatArr` zeigt nun auf die 2
- `*myStatArr` gibt 2
- `myStatArr[2]` gibt 8
- `myStatArr[1] = -4` setzt 3 auf -4

Dynamisch allozierterer Array

Arrays

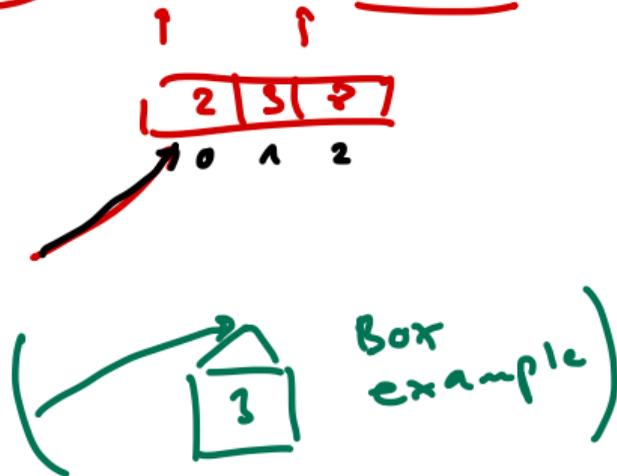
Statisch allozierter Array

```
int myStatArr[3] = {2, 3, 8};
```

- myStatArr zeigt nun auf die 2
- *myStatArr gibt 2
- myStatArr[2] gibt 8
- myStatArr[1] = -4 setzt 3 auf -4

Dynamisch allozierterer Array

```
int* myDynArr = new int[3]{2, 3, 8};
```



Arrays

Statisch allozierter Array

```
int myStatArr[3] = {2, 3, 8};
```

- `myStatArr` zeigt nun auf die 2
- `*myStatArr` gibt 2
- `myStatArr[2]` gibt 8
- `myStatArr[1] = -4` setzt 3 auf -4

Dynamisch allozierterer Array

```
int* myDynArr = new int[3]{2, 3, 8};
```

- `myDynArr` zeigt nun auf die

Arrays

Statisch allozierter Array

```
int myStatArr[3] = {2, 3, 8};
```

- `myStatArr` zeigt nun auf die 2
- `*myStatArr` gibt 2
- `myStatArr[2]` gibt 8
- `myStatArr[1] = -4` setzt 3 auf -4

Dynamisch allozierterer Array

```
int* myDynArr = new int[3]{2, 3, 8};
```

- `myDynArr` zeigt nun auf die 2
- `*myDynArr` gibt

Arrays

Statisch allozierter Array

```
int myStatArr[3] = {2, 3, 8};
```

- `myStatArr` zeigt nun auf die 2
- `*myStatArr` gibt 2
- `myStatArr[2]` gibt 8
- `myStatArr[1] = -4` setzt 3 auf -4

Dynamisch allozierterer Array

```
int* myDynArr = new int[3]{2, 3, 8};
```

- `myDynArr` zeigt nun auf die 2
- `*myDynArr` gibt 2
- `myDynArr[2]` gibt

Arrays

Statisch allozierter Array

```
int myStatArr[3] = {2, 3, 8};
```

- `myStatArr` zeigt nun auf die 2
- `*myStatArr` gibt 2
- `myStatArr[2]` gibt 8
- `myStatArr[1] = -4` setzt 3 auf -4

Dynamisch allozierterer Array

```
int* myDynArr = new int[3]{2, 3, 8};
```

- `myDynArr` zeigt nun auf die 2
- `*myDynArr` gibt 2
- `myDynArr[2]` gibt 8
- `myDynArr[1] = -4`

Arrays

Statisch allozierter Array

```
int myStatArr[3] = {2, 3, 8};
```

- `myStatArr` zeigt nun auf die 2
- `*myStatArr` gibt 2
- `myStatArr[2]` gibt 8
- `myStatArr[1] = -4` setzt 3 auf -4

Dynamisch allozierterer Array

```
int* myDynArr = new int[3]{2, 3, 8};
```

- `myDynArr` zeigt nun auf die 2
- `*myDynArr` gibt 2
- `myDynArr[2]` gibt 8
- `myDynArr[1] = -4` setzt 3 auf -4

Side Note: Arrays are weeeeird

Side Note: Arrays are weeeird

0x7117--

```
int myStatArr[3] = {2, 3, 8};          int* myDynArr = new int[3]{2, 3, 8};

std::cout << &myStatArr[0] << "\t &myStatArr[0] \n";
std::cout << myStatArr << "\t myStatArr \n";
std::cout << &myStatArr << "\t &myStatArr \n";
std::cout << &myDynArr[0] << "\t &myDynArr[0] \n";
std::cout << myDynArr << "\t myDynArr \n";
std::cout << &myDynArr << "\t &myDynArr \n\n\n";
std::cout << typeid(&myStatArr[0]).name() << "\tType of &myStatArr[0]\n";
std::cout << typeid(myStatArr).name() << "\tType of myStatArr\n";
std::cout << typeid(&myStatArr).name() << "\tType of &myStatArr\n";
std::cout << typeid(&myDynArr[0]).name() << "\tType of &myDynArr[0]\n";
std::cout << typeid(myDynArr).name() << "\tType of myDynArr\n";
std::cout << typeid(&myDynArr).name() << "\tType of &myDynArr\n";
```

0x7115--

"

"

Side Note: Arrays are weeeeird

Side Note: Arrays are weeeeird

```
0x7ffcb4fe1d14  &myStatArr[0]
0x7ffcb4fe1d14  myStatArr
0x7ffcb4fe1d14  &myStatArr
0x2340fc0       &myDynArr[0]
0x2340fc0       myDynArr
0x7ffcb4fe1d08  &myDynArr
```

```
Pi      Type of &myStatArr[0]
A3_i    Type of myStatArr
PA3_i   Type of &myStatArr
Pi      Type of &myDynArr[0]
Pi      Type of myDynArr
PPi     Type of &myDynArr
```

delete VS delete []

- Es gilt weiterhin: zu jedem `new` ein `delete`

delete VS delete []

- Es gilt weiterhin: zu jedem **new** ein **delete**
- **delete []** ist der entsprechende Operator zu **new []**

delete VS delete []

- Es gilt weiterhin: zu jedem `new` ein `delete`
- `delete []` ist der entsprechende Operator zu `new []`
- Auch hier aufpassen: Wir löschen nicht den Pointer, sondern `die Range an Objekten`, auf die der Pointer zeigt

delete VS delete []

- Es gilt weiterhin: zu jedem `new` ein `delete`
- `delete []` ist der entsprechende Operator zu `new []`
- Auch hier aufpassen: Wir löschen nicht den Pointer, sondern die Range an Objekten, auf die der Pointer zeigt
- **Häufige Fehlerquelle**
der Aufruf von `delete` für das erste Element, aber nicht für das gesamte Array (mit `delete []`)

Pointer-Arithmetik

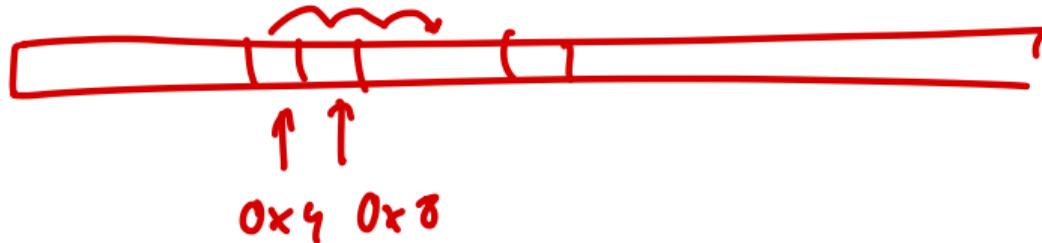
- Wir können mit Pointern "rechnen"
- Die wichtigsten Befehle sind:

Pointer-Arithmetik

- Wir können mit Pointern "rechnen"
- Die wichtigsten Befehle sind:
- Temporäre Shifts

`ptr + 3;`
`ptr - 3;`

*int**
`0x711...002`
 $\frac{1}{2}$



Pointer-Arithmetik

- Wir können mit Pointern "rechnen"
- Die wichtigsten Befehle sind:

- Temporäre Shifts

```
ptr + 3
```

```
ptr - 3
```

- Permanente Shifts

```
ptr++
```

```
--ptr
```

```
ptr += 2
```

Pointer-Arithmetik

- Wir können mit Pointern "rechnen"
- Die wichtigsten Befehle sind:

- Temporäre Shifts

`ptr + 3`

`ptr - 3`

- Permanente Shifts

`ptr++`

`--ptr`

`ptr += 2`

- Distanz zwischen Pointern bestimmen

`ptr_1 - ptr_2`

ptr 1 > ptr 2

Pointer-Arithmetik

- Wir können mit Pointern "rechnen"
- Die wichtigsten Befehle sind:

- Temporäre Shifts

`ptr + 3`

`ptr - 3`

- Permanente Shifts

`ptr++`

`--ptr`

`ptr += 2`

- Distanz zwischen Pointern bestimmen

`ptr_1 - ptr_2`

- Positionen vergleichen

`ptr_1 < ptr_2`

`ptr_1 != ptr_2` ←



Fragen/Unklarheiten?

5. Pointers

5.1. Example: Pointers on Arrays

Pointer-Arithmetik

```
int* a = new int[5]{0, 8, 7, 2, -1};
int* ptr = a;                               // pointer assignment
++ptr;                                       // shift to the right
int my_int = *ptr;                           // read target
ptr += 2;                                    // shift by 2 elements
    // ^ Note how this does not simply "add 2" to the
    // underlying memory address, but instead adds the
    // appropriate amount to get to the integer variable
    // that is stored "2 ints further away"
*ptr = 18;                                   // overwrite target
int* past = a+5;
std::cout << (ptr < past) << "\n";        // compare pointers
```

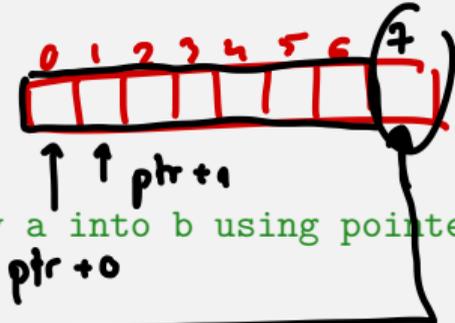
Bug Hunt

Finde und behebe mindestens 3 Probleme in folgendem Programm

```
int* a = new int[7]{0, 6, 5, 3, 2, 4, 1};  
int* b = new int[7];  
int* c = b;
```

```
for (int* p = a; p < a+7; ++p) { // copy a into b using pointers  
    *c++ = *p;  
}
```

```
for (int i = 0; i < 7; ++i) { // cross-check with random access  
    if (a[i] != c[i]) {  
        std::cout << "Oops, copy error...\n";  
    }  
}
```



$c[0] = *c$
 $c[i] = *(c+i)$
 $c[i] = *(c+i)$

Fragen/Unklarheiten?

5. Pointers

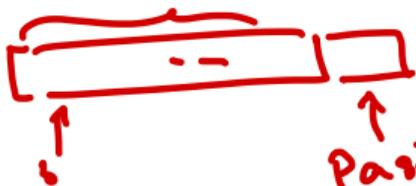
5.2. Example: Special Copy

Special Copy?

```
// PRE: [b, e) and [o, o+(e-b)) are disjoint valid ranges
// POST: - - - - - TODO: determine it! - - - - -
//      - - - - -
void f (int* b, int* e, int* o) {
    while (b != e) {
        --e;
        *o = *e;
        ++o;
    }
}
```

$pte_ptr = first_ptr + count$

$[b, e)$ ← offer



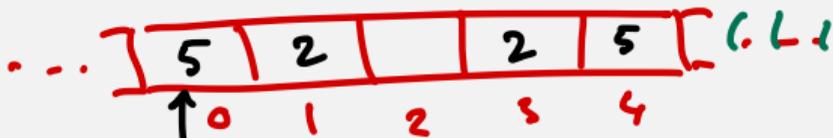
Part the End

Reverse Copy!

```
// PRE: [b, e) and [o, o+(e-b)) are disjoint valid ranges
// POST: The range [b, e) is copied in reverse order
//       into the range [o, o+(e-b))
void f (int* b, int* e, int* o) {
    while (b != e) {
        --e;
        *o = *e;
        ++o;
    }
}
```

Reverse Copy!

```
// PRE: [b, e) and [o, o+(e-b)) are disjoint valid ranges
// POST: The range [b, e) is copied in reverse order
//       into the range [o, o+(e-b))
void f (int* b, int* e, int* o) {
    while (b != e) {
        --e;
        *o = *e;
        ++o;
    }
}
```

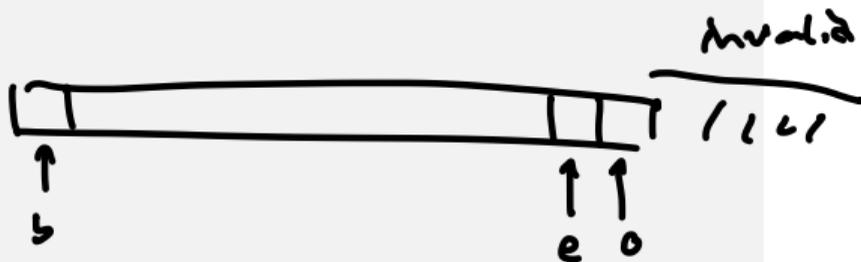


Welche dieser Eingaben sind valid nach `int* a = new int[5];?`

- a) `f(a, a+5, a+5)` b) `f(a, a+2, a+3)` c) `f(a, a+3, a+2)`

Reverse Copy!

```
// PRE: [b, e) and [o, o+(e-b)) are disjoint valid ranges
// POST: The range [b, e) is copied in reverse order
//        into the range [o, o+(e-b))
void f (int* b, int* e, int* o) {
    while (b != e) {
        --e;
        *o = *e;
        ++o;
    }
}
```



Welche dieser Eingaben sind valid nach `int* a = new int[5];`?

a) `f(a, a+5, a+5)` b) `f(a, a+2, a+3)` c) `f(a, a+3, a+2)`

Antwort: b)

Fragen/Unklarheiten?

Pointer Constness

Es gibt zwei Arten von Constness bei Pointern:

```
const int* ptr = &a;
```

Pointer Costness

const



Es gibt zwei Arten von Constness bei Pointern:

```
(const int)* ptr = &a;
```

kein Schreibzugriff auf a

Pointer Constness

Es gibt zwei Arten von Constness bei Pointern:

```
const int* ptr = &a;
```

kein Schreibzugriff auf `a`
d.h. wir dürfen den Wert des
Integers `a` *nicht* verändern

Pointer Costness

Es gibt zwei Arten von Constness bei Pointern:

```
const int* ptr = &a;
```

kein Schreibzugriff auf a
d.h. wir dürfen den Wert des
Integers a *nicht* verändern

```
int* const ptr = &a;
```



Pointer Costness

Es gibt zwei Arten von Constness bei Pointern:

```
const int* ptr = &a;
```

kein Schreibzugriff auf `a`
d.h. wir dürfen den Wert des
Integers `a` *nicht* verändern

```
int* const ptr = &a;
```

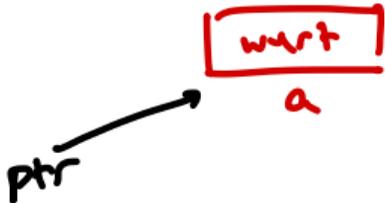
kein Schreibzugriff auf `ptr`

Pointer Costness

Es gibt zwei Arten von Constness bei Pointern:

```
const int* ptr = &a;
```

kein Schreibzugriff auf `a`
d.h. wir dürfen den Wert des
Integers `a` *nicht* verändern



```
int* const ptr = &a;
```

kein Schreibzugriff auf `ptr`
d.h. wir dürfen nicht ändern,
wohin der Pointer zeigt



Fragen/Unklarheiten?

6. Aufgabe "Push Back"

Aufgabe "Push Back"

- Öffnet "Push Back" auf **code expert**

Aufgabe "Push Back"

- Öffnet "Push Back" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet

Aufgabe "Push Back"

- Öffnet "Push Back" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet
- Programmiert eine Lösung (optional in Gruppen)

Lösung zu "Push Back"

```
// PRE:  source_begin points to first element to be copied;
//       source_ends points to element after the last element to be copied;
//       destination_begin points to first element of target memory block;
//       #elements in target memory location >= #elements in source;
// POST: copies the content of the source memory block to the destination
//       memory block.
void copy_range(const int* const source_begin,
               const int* const source_end,
               int* const destination_begin  ){

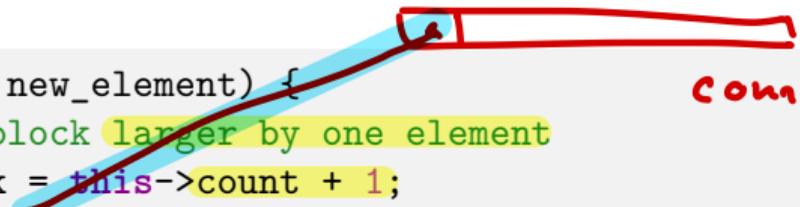
    int* dst = destination_begin;
    for (const int* src = source_begin; src != source_end; ++src) {
        *dst = *src;
        ++dst;
    }
}
```

Lösung zu "Push Back"

```
void our_vector::push_back(int new_element) {  
    // 1. Allocate a new memory block larger by one element  
unsigned int lenghtOfNewBlock = this->count + 1;  
    int* const ptrToNewBlock = new int[lenghtOfNewBlock];  
  
    // 2. Copy all the elements from the old memory block to the new one  
    copy_range(this->elements, this->elements + count, ptrToNewBlock);  
  
    // 3. Deallocate the old memory block  
    delete[] this->elements; // frees memory from old elements  
    → this->elements = ptrToNewBlock; // redirects pointer to new block  
  
    // 4. Add the new element at the end of the new memory block  
    this->elements[count] = new_element;  
    count++; // increment counter  
}
```

count + 1

ptr von elements

A diagram consisting of a long horizontal rectangle representing a memory block. A red box highlights the rightmost portion of this rectangle. A red arrow points from the center of this highlighted box to the right, extending beyond the main rectangle. A blue diagonal line is drawn from the top-left corner of the main rectangle towards the tip of the red arrow.

Fragen/Unklarheiten?

7. Memory Management

```
// PRE: len is the length of the memory block that starts at array
void test1(int* array, int len) {
    int* fourth = array + 3;
    if (len > 3) {
        std::cout << *fourth << std::endl;
    }
    for (int* p = array; p != array + len; ++p) {
        std::cout << *p << std::endl;
    }
}
```

Finde Fehler im Code und schlage Korrekturen vor

Bug Hunt I — Dangerous Pointer

```
// PRE: len is the length of the memory block that starts at array
void test1(int* array, int len) {
    //int* fourth = array + 3;    // ERROR
    if (len > 3) {
        int* fourth = array + 3;    // OK
        std::cout << *fourth << std::endl;
    }
    for (int* p = array; p != array + len; ++p) {
        std::cout << *p << std::endl;
    }
}
```

Auch wenn der Pointer nicht dereferenziert wird, muss er in einen Speicherblock oder auf das Element unmittelbar nach dessen Ende zeigen.

Bug Hunt II

```
// PRE: len >= 2
int* fib(int len) {
    int* array = new int[len];
    array[0] = 0; array[1] = 1;
    for (int* p = array+2; p < array + len; ++p) {
        *p = *(p-2) + *(p-1); }
    return array; }
void print(int* array, int len) {
    for (int* p = array+2; p < array + len; ++p) {
        std::cout << *p << " ";
    }
}
void test2(int len) {
    int* array = fib(len);
    print(array, len);
}
```

Bug Hunt II — Memory Leak

```
// PRE: len >= 2
int* fib(int len) {
    int* array = new int[len];
    array[0] = 0; array[1] = 1;
    for (int* p = array+2; p < array + len; ++p) {
        *p = *(p-2) + *(p-1); }
    return array; }

void print(int* array, int len) {
    for (int* p = array+2; p < array + len; ++p) {
        std::cout << *p << " ";
    }
}

void test2(int len) {
    int* array = fib(len);
    print(array, len);
    delete[] array;           // otherwise array is leaked!
}
```

Bug Hunt III

```
// PRE: len >= 2
int* fib(int len) {
    // ...
}
void print(int* m, int len) {
    for (int* p = m+2; p < m + len; ++p) {
        std::cout << *p << " ";
    }
    delete m;
}
void test2(int len) {
    int* array = fib(len);
    print(array, len);
    delete[] array;
}
```

Bug Hunt III — Double Free!

```
// PRE: len >= 2
int* fib(int len) {
    // ...
}
void print(int* m, int len) {
    for (int* p = m+2; p < m + len; ++p) {
        std::cout << *p << " ";
    }
    delete[] m;
}
void test2(int len) {
    int* array = fib(len);
    print(array, len);
    // delete[] array;           // array deallocated twice!
}
```

Fragen/Unklarheiten?

8. Muddiest Point

Woran hakt es bei dir?

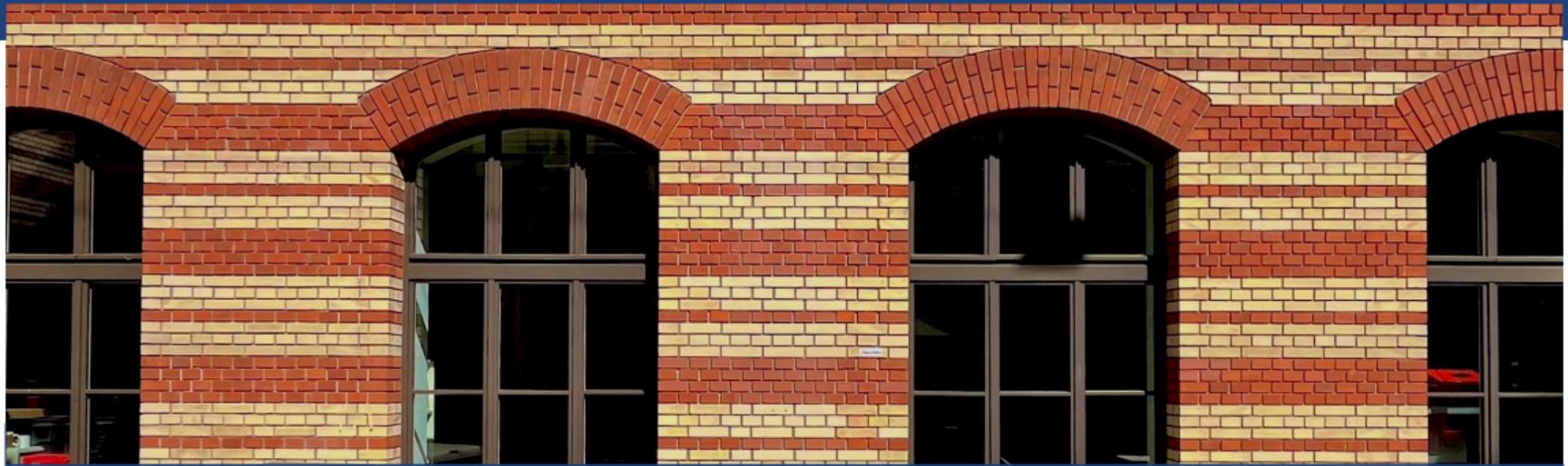
(Absichtlich leer gelassen)

9. Outro

Allgemeine Fragen?

Bis zum nächsten Mal

Schöne Woche noch!



Exercise Session — Computer Science — 13

Adel Gavranović

Rätsel, Knackpunkt(e), Fragerunde

Übersicht

Follow-up

Memory Management / Bug Hunt

Alte Prüfungen

Riddle

(Prüfungsvorbereitung)

Knackpunkte

if-Klauseln vereinfachen



`n.ethz.ch/~agavranovic`

 Material

 Webpage

 Mail

1. Intro

Intro

- Letzte Übungsstunde gemeinsam :(

2. Follow-up

2. Follow-up

2.1. Memory Management / Bug Hunt

Bug Hunt I

```
// PRE: len is the length of the memory block that starts at array
void test1(int* array, int len) {
    int* fourth = array + 3;
    if (len > 3) {
        std::cout << *fourth << std::endl;
    }
    for (int* p = array; p != array + len; ++p) {
        std::cout << *p << std::endl;
    }
}
```

Finde Fehler im Code und schlage Korrekturen vor

Bug Hunt I — Dangerous Pointer

```
// PRE: len is the length of the memory block that starts at array
void test1(int* array, int len) {
    //int* fourth = array + 3;    // ERROR
    if (len > 3) {
        int* fourth = array + 3;    // OK
        std::cout << *fourth << std::endl;
    }
    for (int* p = array; p != array + len; ++p) {
        std::cout << *p << std::endl;
    }
}
```

Auch wenn der Pointer nicht dereferenziert wird, muss er in einen Speicherblock oder auf das Element unmittelbar nach dessen Ende zeigen.

Bug Hunt II

```
// PRE: len >= 2
int* fib(int len) {
    int* array = new int[len];
    array[0] = 0; array[1] = 1;
    for (int* p = array+2; p < array + len; ++p) {
        *p = *(p-2) + *(p-1); }
    return array; }

void print(int* array, int len) {
    for (int* p = array+2; p < array + len; ++p) {
        std::cout << *p << " ";
    }
}

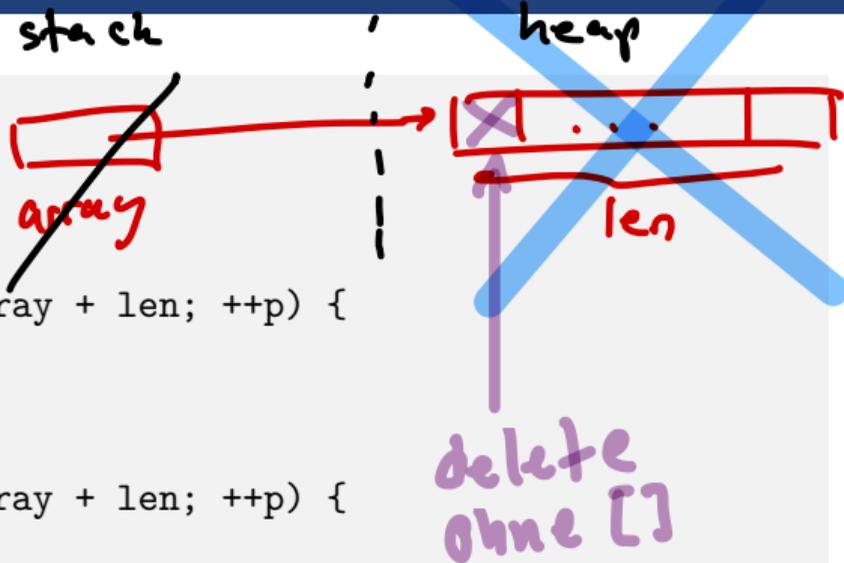
void test2(int len) {
    int* array = fib(len);
    print(array, len);
}
```

Bug Hunt II – Memory Leak

```
// PRE: len >= 2
int* fib(int len) {
    int* array = new int[len];
    array[0] = 0; array[1] = 1;
    for (int* p = array+2; p < array + len; ++p) {
        *p = *(p-2) + *(p-1); }
    return array; }

void print(int* array, int len) {
    for (int* p = array+2; p < array + len; ++p) {
        std::cout << *p << " ";
    }
}

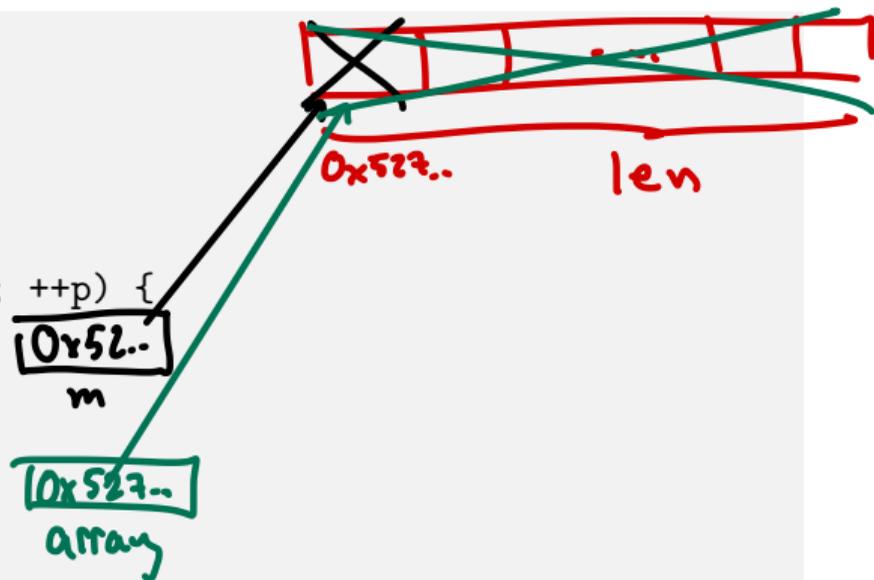
void test2(int len) {
    int* array = fib(len);
    print(array, len);
    delete[] array; // otherwise array is leaked!
}
```



Bug Hunt III

```
// PRE: len >= 2
int* fib(int len) {
    // ...
}
void print(int* m, int len) {
    for (int* p = m+2; p < m + len; ++p) {
        std::cout << *p << " ";
    }
    delete m;
}
void test2(int len) {
    int* array = fib(len);
    print(array, len);
    delete[] array;
}
```

somewhere in memory!



Bug Hunt III — Double Free!

```
// PRE: len >= 2
int* fib(int len) {
    // ...
}
```

```
void print(int* m, int len) {
    for (int* p = m+2; p < m + len; ++p) {
        std::cout << *p << " ";
    }
    delete[] m;
```

```
}
void test2(int len) {
    int* array = fib(len);
    print(array, len);
    // delete[] array; // array deallocated twice!
}
```

* would be more elegant
to call delete[] in test2
and not in print

um das gesamte array zu deallocaten

Fragen/Unklarheiten?

3. Lernziele

Ziele

- Wissen, wo man alte Prüfungen findet und mit ihnen üben kann
- Knackpunkte geklärt oder zumindest ~~richte~~ richtige Ressourcen gefunden

Ziele

- Wissen, wo man alte Prüfungen findet und mit ihnen üben kann
- Knackpunkte geklärt oder zumindest richtige Ressourcen gefunden
- Sich für das heutige *rw::gettogether* angemeldet

4. Feedback zu **code** expert

"Rekursive Lösungen"

¹keine Garantien meinerseits

"Rekursive Lösungen"

- Viele haben in ihren Lösungen zu den "rekursiven" Aufgaben eine zweite Funktion eingeführt welche einen Teil der Rekursion für Spezialfälle übernimmt. Das ist suboptimal und nicht "richtig" rekursiv

¹keine Garantien meinerseits

"Rekursive Lösungen"

- Viele haben in ihren Lösungen zu den "rekursiven" Aufgaben eine zweite Funktion eingeführt welche einen Teil der Rekursion für Spezialfälle übernimmt. Das ist suboptimal und nicht "richtig" rekursiv
- Studiert nach Abgabe der Rekursionsaufgaben unbedingt die Musterlösungen. Sie sind oft sehr elegant implementiert und erlauben es, sie auf ähnliche Aufgabenstellungen zu übertragen

¹keine Garantien meinerseits

"Rekursive Lösungen"

- Viele haben in ihren Lösungen zu den "rekursiven" Aufgaben eine zweite Funktion eingeführt welche einen Teil der Rekursion für Spezialfälle übernimmt. Das ist suboptimal und nicht "richtig" rekursiv
- Studiert nach Abgabe der Rekursionsaufgaben unbedingt die Musterlösungen. Sie sind oft sehr elegant implementiert und erlauben es, sie auf ähnliche Aufgabenstellungen zu übertragen
- Das Muster, das wir gelernt und geübt haben (Base Case und dann Recursive Case mit Leap of Faith) ist praktisch immer¹ anwendbar

¹keine Garantien meinerseits

5. Alte Prüfungen

Mit alten Prüfungen üben

Mit alten Prüfungen üben

- Auf der Website des Kurses sind zahlreiche alte Prüfungen zu finden

Mit alten Prüfungen üben

- Auf der Website des Kurses sind zahlreiche alte Prüfungen zu finden
- Sie können eine gute Möglichkeit sein, für die Prüfung zu üben

Mit alten Prüfungen üben

pw: Informatik

- Auf der Website des Kurses sind zahlreiche alte Prüfungen zu finden
- Sie können eine gute Möglichkeit sein, für die Prüfung zu üben
- Stellt sicher, dass ihr die richtigen anschaut (die für euren Kurs)



`lec.inf.ethz.ch/past_exams`

6. Riddle

Riddle

Riddle me dieses Codebeispiel auf **code expert** ...

7. Prüfungsvorbereitung

- **Wissenslücken Stopfen**

■ Wissenslücken Stopfen

- findet schnellstmöglich raus, wo Ihr Wissenslücken/Skill-issues habt und geht sie an (ggf. zusammen mit anderen)

■ **Wissenslücken Stopfen**

- findet schnellstmöglich raus, wo Ihr Wissenslücken/Skill-issues habt und geht sie an (ggf. zusammen mit anderen)

■ **Alte Prüfungen Lösen**

■ Wissenslücken Stopfen

- findet schnellstmöglich raus, wo Ihr Wissenslücken/Skill-issues habt und geht sie an (ggf. zusammen mit anderen)

■ Alte Prüfungen Lösen

- in Prüfungssetting (@MacUsers: gewöhnt euch ans Keyboard!)

■ Wissenslücken Stopfen

- findet schnellstmöglich raus, wo Ihr Wissenslücken/Skill-issues habt und geht sie an (ggf. zusammen mit anderen)

■ Alte Prüfungen Lösen

- in Prüfungssetting (@MacUsers: gewöhnt euch ans Keyboard!)
- letzten zwei spart man sich für die letzte Woche vor der Prüfung auf

■ Wissenslücken Stopfen

- findet schnellstmöglich raus, wo Ihr Wissenslücken/Skill-issues habt und geht sie an (ggf. zusammen mit anderen)

■ Alte Prüfungen Lösen

- in Prüfungssetting (@MacUsers: gewöhnt euch ans Keyboard!)
- letzten zwei spart man sich für die letzte Woche vor der Prüfung auf

■ Zusammenfassung verfassen, mit...

■ Wissenslücken Stopfen

- findet schnellstmöglich raus, wo Ihr Wissenslücken/Skill-issues habt und geht sie an (ggf. zusammen mit anderen)

■ Alte Prüfungen Lösen

- in Prüfungssetting (@MacUsers: gewöhnt euch ans Keyboard!)
- letzten zwei spart man sich für die letzte Woche vor der Prüfung auf

■ Zusammenfassung verfassen, mit...

- Auswertungsreihenfolgen,
- komplizierte Syntax,
- Programmieraufgaben, mit denen ihr sehr viel Mühe hattet

■ Wissenslücken Stopfen

- findet schnellstmöglich raus, wo Ihr Wissenslücken/Skill-issues habt und geht sie an (ggf. zusammen mit anderen)

■ Alte Prüfungen Lösen

- in Prüfungssetting (@MacUsers: gewöhnt euch ans Keyboard!)
- letzten zwei spart man sich für die letzte Woche vor der Prüfung auf

■ Zusammenfassung verfassen, mit...

- Auswertungsreihenfolgen,
- komplizierte Syntax,
- Programmieraufgaben, mit denen ihr sehr viel Mühe hattet

■ weiteres?

8. Knackpunkte

Woran hakt es bei dir?

Pointers (basics)

(Absichtlich leer gelassen)

$0x525..$ ← address
7 ← value
 a ← name
 int ← type

$0x7AB..$ ← address
 $0x525..$ ← value
 a_ptr ← name
 int* ← type

$0x127..$ ← address
val | ... | f | ptr ← "value"
 b ← name
 Box ← "type"

$0x42A..$ ← address
 $0x129..$ ← value
 b_ptr ← name
 Box* ← type

```

class Box{
  int val;
  ;
  void fl..
  };
  
```

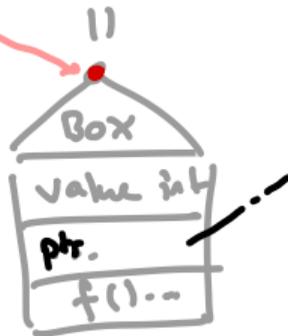
$0x775..$
42
 int

// in code

```

int* a_ptr = da;
int* A_ptr = new int(42);
           "0x775..
  
```

class
 ↓
 Box ≠ b
 object
 ↓
 instances of
 a given class



(Absichtlich leer gelassen)

9. if-Klauseln vereinfachen

Simplere ifs I

Wir haben festgestellt, dass viele solchen Code schreiben

Simpleere ifs I

Wir haben festgestellt, dass viele solchen Code schreiben

```
if (grid != nullptr) {
    if (grid->is_filled(row, col)) {
        if (col == 8) {
            if (fillValidNumber(grid, row + 1, 0)) {
                return true;
            }
        } else {
            if(fillValidNumber(grid, row, col + 1)) {
                return true;
            }
        }
        return false;
    }
}
```

Simplere ifs II

Da die Funktion `fillValidNumber` einen booleschen Wert liefert und wir auch einen Booleschen Wert zurückgeben wollen, können wir die verschachtelte `if`-Klausel und das letzte `return` entfernen

Simpleere ifs II

Da die Funktion `fillValidNumber` einen booleschen Wert liefert und wir auch einen Booleschen Wert zurückgeben wollen, können wir die verschachtelte `if`-Klausel und das letzte `return` entfernen

```
if (grid != nullptr) {
    if (grid->is_filled(row, col)) {
        if (col == 8) {
            return fillValidNumber(grid, row + 1, 0);
        } else {
            return fillValidNumber(grid, row, col + 1);
        }
    }
}
```

Simplere ifs III

Wir können auch die Tatsache nutzen, dass `&&` einen Kurzschluss darstellt, um die die beiden äusseren `if`-Anweisungen zusammenzuführen

Simplere ifs III

Wir können auch die Tatsache nutzen, dass `&&` einen Kurzschluss darstellt, um die die beiden äusseren `if`-Anweisungen zusammenzuführen

```
if (grid != nullptr && grid->is_filled(row, col)) {  
    if (col == 8) {  
        return fillValidNumber(grid, row + 1, 0);  
    } else {  
        return fillValidNumber(grid, row, col + 1);  
    }  
}
```

if (condition
 se am 2
 se am 3
 am 4) {

f(...
 == } {

>

Fragen/Unklarheiten?

10. Outro

Allgemeine Fragen?

Bis zum nächsten Mal

Bis heute Abend und viel Erfolg bei den Prüfungen!