



Übungsstunde W09

Informatik (RW & CBB & Statistik) – HS 23

Heutiges Programm

Follow-up

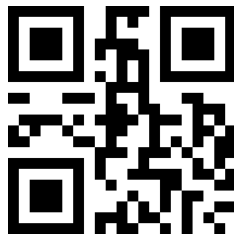
Feedback zu **code expert**

Ziele

Structs

Rekursion II

Outro



rwko.ch/lily

1. Follow-up

Follow-up aus vorherigen Übungsstunden

- Effizienz von "**read_matrix**":
 - Die Version mit der vorher initialisierten Matrix sollte im Allgemeinen effizienter sein, da weniger dynamisch allozierter Speicherplatz benötigt wird¹
 - Es macht erst bei sehr grossen Matrizen wirklich einen Unterschied, ihr müsst euch also (noch) keine Sorgen darüber machen

¹Ihr werdet gegen Ende des Semesters lernen, was das genau heisst

2. Feedback zu **code** expert

Allgemeines zu **code expert**

Ein paar Vereinfachungen für euren Code²

```
if(condition == true){  
    // ...  
}
```

→

```
if(condition){  
    //...  
}
```

```
if(condition){  
    return true;  
} else {  
    return false;  
}
```

→

```
return condition;
```

in Funktionen, die einen **bool** zurückgeben

²Nicht vergessen: Vereinfachungen sind nicht immer besser für die Verständlichkeit

E7:T1: "Const and reference types"

- Was heisst **const**?
 - Sobald eine **cpp** Variable initialisiert ist, darf ihr Wert nicht mehr verändert werden
 - Die Variable darf im Programm benutzt werden ("Lesezugriff")
- Wann wird constness (nicht) respektiert?
 - Default: ist nichts als **const** deklariert, so ist die constness respektiert
 - Sonst: Es darf nicht versucht werden, den Wert einer **const** Variable zu verändern (kein Schreibzugriff)

Fragen/Unklarheiten?

3. Ziele

Ziele

- Structs definieren und benutzen können
- Schwierigere Probleme, die Rekursion beinhalten, lösen können

4. Structs

Ein `struct` ist ein Bündel von Zeugs

- Das können Variablen, Funktionen, weitere structs und viel mehr sein ("Members")
- Typen müssen nicht dieselben sein
- Bieten uns eine Möglichkeit neue "Objekte" zu definieren, z.B. einen eigenen Zahlentypen oder mathematische Objekte wie Geraden, Quadrate, Kreise, etc.
- Wichtig: `;` am Schluss der Definition nicht vergessen

Struktur von struct

```
struct Person {
    unsigned int age;
    std::string field;
    std::vector<int> lucky_nums;
};

int main () {
    Person Adel = {26, "Computer Science", {42, 161}};
    Person Deli = Adel;
    Person Jules = {25, "Linguistics", {13, 12}};
    Person Lily = {19, "Computational Science", {9, 19}};
    std::cout << "Adel's " << Adel.age <<
                " years old" << std::endl;
    return 0;
}
```

Struktur von struct

```
struct Person {
    unsigned int age;
    std::string field;
    std::vector<int> lucky_nums;
    void function(std::string str){
        std::cout << str;
    }
};

int main () {
    Person Adel = {26, "Computer Science", {42, 161}};
    Person Deli = Adel;
    Person Jules = {25, "Linguistics", {13, 12}};
    Person Lily = {19, "Computational Science", {9, 19}};
    Lily.function("hello, world!");
    return 0;
}
```

Fragen/Unklarheiten?

Aufgabe "Geometry Exercise"

- Öffnet "Geometry Exercise" auf **code expert**
- Überlegt euch, wie ihr das Problem mit Stift und Papier angehen würdet
- Programmiert eine Lösung (optional in Gruppen)

Lösung zu "Geometry Exercise"

```
// Subtask 1: adding vectors
// POST: returns the sum of a and b
vec sum(const vec& a, const vec& b) {
    // version 1: compact, used for the rest of the example
    return {a.x + b.x, a.y + b.y, a.z + b.z};

    // version 2: longer but maybe easier to understand
    // vec tmp;
    // tmp.x = a.x + b.x;
    // tmp.y = a.y + b.y;
    // tmp.z = a.z + b.z;
    // return tmp;
}
```

Lösung zu "Geometry Exercise"

```
// Subtask 2: defining a line in 3D
struct line {
    vec start;
    vec end; // INV: start != end
};
// helper function to print a vector
void print_line(const line& l) {
    print_vec(l.start);
    std::cout << " <-> ";
    print_vec(l.end);
}
```

Lösung zu "Geometry Exercise"

```
// Subtask 3: shifting line by a vector
// POST: returns a new line obtained by shifting l
// by v.
line shift_line(const line& l, const vec& v) {
    return {sum(l.start, v), sum(l.end, v)};
}

// Subtask 4: overloading the + operator for vectors
vec operator+(const vec& a, const vec& b) {
    return sum(a, b);
}
```

Lösung zu "Geometry Exercise"

```
// Subtask 5: overloading the + operator for lines
// version 1: use the shift_line function
line operator+(const line& l, const vec& v) {
    return shift_line(l, v);
}

// version 2: make use of the overloaded + operator for vectors
line operator+(const line& l, const vec& v) {
    return {l.start + v, l.end + v};
}
```

Fragen/Unklarheiten?

5. Rekursion II

Aufgabe Power Set

DiskMath Recap

- Ein Potenzmenge ist die Menge aller Teilmengen
- $\mathcal{P}(S) := \{X \mid X \subseteq S\}$
- Beispiel:
 - Gegeben sei die Menge $A = \{a, b, c\}$
 - Die Potenzmenge ist
 $\mathcal{P}(A) = \{\{\}, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$

Einführung zu set.h

- `set` ist eine selbstgebastelter Typ! (ein `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**
- 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?

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 subset³

$$\mathcal{P}(\{b, c, d\}) = \{\{\}, \{b\}, \{c\}, \{d\}, \{b, c\}, \dots\}$$

// 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

$$\left\{ \begin{array}{l} \{\}, \{b\}, \{c\}, \{d\}, \{b, c\}, \dots, \\ \{a\}, \{a, b\}, \{a, c\}, \{a, d\}, \{a, b, c\}, \dots, \end{array} \right\}$$

³Hier kommt der Vertrauensvorschuss der Rekursion (*Recursive Leap of Faith*) zum Tragen

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?

Towers of Hanoi

Everyone: it's a game for kids



Programmers:



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)

Fragen/Unklarheiten?

6. Outro

Allgemeine Fragen?

Bis zum nächsten Mal

Schöne Woche!