

# Übungsstunde

## Woche 07

Adel Gavranović  
adel.gavranovic@inf.ethz.ch

# Overview

## Heutige Themen

Intro

Referenzen

Characters

Outro

## Links

▶ [polybox zum Material für die Übungsstunden](#)

▶ [Mail an Assistenten](#)

# Handout



# Follow-up aus vorheriger Übungsstunde

- *Variablen ausserhalb von Funktionen*
  - Variablen sollten *möglichst nie* ausserhalb von Funktionen initialisiert/definiert werden. Begründung: Kann zu sehr(!) doofen Bugs beim *Linking* (nicht prürel) führen.

# Kommentare zu [code] expert

## Allgemein

- Sehr viele sehr gut formatierte Kommentare, weiter so! :)

## E5:T3

- Die meisten Submissions waren gut aber ich empfehle euch allen die ML anzuschauen, da sie um einiges eleganter implementiert ist
- Eine Funktion konvertiert was auch immer returned wird automatisch zum return type

## E5:T4

- PRE- und POST-Conditions können auch einfach  $0 < x < 2$  lauten. Hauptsache man versteht's und es sieht gut aus

# Sehr coole Lösung<sup>1</sup> zu E5:T3

```
int round(double x) {  
    x += 0.5 - (x < 0); // add 0.5 to x if positive  
                        // and if negative subtract 0.5  
    int r = x;         // store truncated x in r  
    return r;         // function returns rounded x  
}
```

---

<sup>1</sup>von James Leadbeater

# Fragen zu `[code]` expert eurerseits?

# Lernziele

- Programme, die Referenztypen verwenden, tracen können
- Mit `std::vector` umgehen können (erstellen, modifizieren, ein- und auslesen)
- Mit ASCII in C++ umgehen können



# Beispiel zu Program Tracing I

```
int a = 3;  
int& b = a;
```

```
b = 2;
```

```
std::cout << a;
```

```
// output "2"
```

## 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", but why?
```

Referenzen werden als Typ von Funktionsparameter (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: "5", but why?
```

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

# Exercises

## Wieso das Ganze?

Weil...

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

# Fragen/Unklarheiten?

# 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", but why?
```

# Fragen/Unklarheiten?

## Exercise 2



## Exercise 2

(a)

What is the output of the program for the following variant of `foo`?

```
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;  
}
```

## Exercise 2

(a)

What is the output of the program for the following variant of `foo`?

1 2 4 8 16

```
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;  
}
```

## Exercise 2

(b)

What is the output of the program for the following variant of `foo`?

```
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;  
}
```

## Exercise 2

(b)

What is the output of the program for the following variant of `foo`?

```
1 1 1 1 1
```

```
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;  
}
```

## Exercise 2

(c)

What is the output of the program for the following variant of `foo`?

```
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;  
}
```

## Exercise 2

(c)

What is the output of the program for the following variant of `foo`?

```
1 1 1 1 1
```

```
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;  
}
```

# Exercise "Converting Input to UPPER CASE"

## Aufgabe

Write a program that reads a sequence of characters, delimited by the new-line character, as a vector of chars. Then the program should output the sequence with all lower-case letters changed to UPPER-CASE letters.

To read the sequence you can:

- read a single character from standard input;
- insert it into a vector of chars;
- repeat until you find a new line character ( $\backslash n$ ).

Please put the code that converts the entire sequence to upper-case and a single character to upper-case into separate functions (you should have at least three functions).

Hint: variables of type char can be treated as numbers.

# Exercise "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



## Solution "Converting Input to UPPER CASE"

```
#include <iostream>
#include <vector>
#include <ios>

// POST: Converts the letter to upper case.
void char_to_upper(char& letter){
    if('a' <= letter && letter <= 'z'){
        letter -= 'a' - 'A'; // 'a' > 'A'
    }
}

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

## Solution "Converting Input to UPPER CASE"

```
std::cin >> std::noskipws;

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(unsigned int i = 0; i < letters.size(); ++i){
    std::cout << letters.at(i);
}
```

# How to `std::vector`

- `#include <vector>`
- Vectors kann man sich als eine Reihe von Kästchen vorstellen, die je einen Wert des gegebenen Types speichern
- Man kann Vectors etwa so behandeln wie einen neuen Typen
- `std::vector<int> myvector{1,2,3};`  
um einen Vector zu initialisieren
- Es gibt viele Möglichkeiten einen Vector zu initialisieren/definieren. Schaut in den Summaries nach oder sucht online
- `myvector[n-1]`  
um den  $n$ -ten Wert im Vector `myvector` zu benutzen
- `myvector.push_back(x)`  
um den Wert `x` anzuhängen

# Fragen/Unklarheiten?

# Tipps für [code] expert

## E7:T1 "Const and reference types"

- Zerbrecht euch nicht den Kopf wegen *semantically* oder *runtime* Validity
- Versucht euer bestes und verbringt nicht zu viel Zeit mit dieser Aufgabe

## E7:T2 "Number of Occurrences"

- Aufgabe → Teilprobleme → Funktionen
- Vergesst nicht, Referenzen richtig zu verwenden

## E7:T3 "Longest Increasing Subsequence"

- dito
- Besprecht die Aufgabe mit anderen um einen Guten Ansatz zu finden

## E7:T4 "Sorting by Swapping"

- dito
- Vergesst nicht: Erst XP holen, dann optimieren

# Allgemeine Fragen?

# Bis zum nächsten Mal

- macht eure Hausaufgaben
- bleibt gesund