



# Exercise Session W04

Computer Science (CSE) – AS 23

# Overview

## Today's Agenda

Elephant in the Room

Follow-up

Feedback on **code expert**

Expressions

Loops

Calculating Sums

Tips for **code expert**

Outro



[rwko.ch/lily](https://rwko.ch/lily)

# 1. Elephant in the Room

---

# Where's Adel?

- Adel's stuck in Amsterdam
- will, Deutsche Bahn permitting, be on a train home by now
- Groetjes uit Amsterdam!



## 2. Follow-up

---

# Follow-up from last exercise session

- Pretty clear vote: This exercise session is now taught in English!
- Yay democracy!
  - 22 Votes, of which...
    - ▶ ...40% agreed to the switch to English
    - ▶ ...60% didn't care
- Pardon the many typos to come
- You can still send mails and ask questions in (Swiss) German

### 3. Feedback on **code** expert

---

# General things regarding **code expert**

- All the text based tasks should be marked by now<sup>1</sup>
  - Questions regarding material/task → Mail to TA
  - Questions regarding corrections → Mail to TA
  - Bugs in **code expert** → Mail to Head TA
- Programming tasks still outstanding

---

<sup>1</sup>If you're enrolled in my group on **code expert**



# Objectives

- Be able to evaluate complex expressions (involving arithmetic and booleans)
- Be able to implement and use sums in C++
- Be familiar with all kinds of loops (`for`, `while`, `do-while`) and be able to trace them
- Be able to replace each kind of loop with any other

# Comments regarding **code expert**

Please be aware that your code is going to be read by other people, in particular TAs, and that you should strive to make your code legible and comprehensible.

```
// even small comments  
// can make a big difference
```

# Comments regarding **code expert**

## **Formatting and Structure**

- Use empty lines to separate blocks of code
- Use tabs/spaces to put similar blocks onto the same height
- Don't write further than the small gray line on the right

## **Comments**

- Document your code (in particular if math or tricks are used)
- Put questions/thoughts/approaches at the very top as a comment
- German or English are fine

## **Task Description/Autograder**

- Corrections fairly strict (in the beginning) regarding not following the task description

# Comments regarding **code expert**

## **E2:T1 Expressions**

- Valid expressions don't necessarily need to be saved anywhere

Questions regarding **code expert** ?

## 4. Expressions

---

# Types

## Types covered so far

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

Sometimes, multiple types are present in the same expression.  
How do different types interact?

## Generality order of types

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

Types always convert to the more general type in an expression

# Mental model of types

Type (literal)	Approximates
bool	$\mathbb{B} = \{\text{false}, \text{true}\}$
unsigned int (u)	$\mathbb{N}$
int	$\mathbb{Z}$
float (f)	$\mathbb{R}$
double	$\mathbb{R}$ , but <i>double</i> precision



# Evaluating Types I

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

## Solution

double, 2.5, since the int 2 gets turned into a double 2.0 first in order to calculate this expression.

# Evaluating Types II

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

## Solution

`double`, 0 because the left expression `1/2` gets evaluated first, which evaluates to 0, since it's an integer division. The rest is trivial, since `0*anything` evaluates to 0. That 0 will be of type `double`.

# Literals

There are certain letters which are assigned certain meanings regarding types. If you want to tell the compiler “*Hey, don't treat this 2.0 as a double, but instead as a float*” you have to put an `f` at the end of the value. Like this:

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

# Evaluating Types III

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

## Solution

float, 5.0, can be written as 5.0f.

First, the 5/2 gets evaluated which results in 2 (integer division). Then 2.0f\*5.0f: The int 2 became a float because that is the more general type (in this expression). Ditto for /2 later.

# Exercise I

1. Which of the following character sequences are not C++ expressions, and why not? Here, `x` and `y` are variables of type `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. For all of the valid expressions that you have identified in 1, decide whether these are lvalues or rvalues and explain your decision.
3. Determine the values of the expressions and explain how these values are obtained. Assume that initially `x == 1` and `y == -1`.

# Expression Evaluation - Solutions 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-VALUE

# Expression Evaluation - Solutions b)

`y = (x++ = 3)`

INVALID

# Expression Evaluation - Solutions 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-VALUE



# Expression Evaluation - Solutions 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-VALUE

# Loop Correctness

Can a user of the program observe the difference between the output produced by these three loops? If yes, how? Assume that `n` is a variable of type `unsigned int` whose value is given by the user.

```
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);
```

# Loop Correctness - Solution

## Solution

There are the following differences:

- Unlike loops 1 and 2, loop 3 does output `|1|` for input `|n == 0|` because the statement in a `|do|-loop` is always executed once before the condition is checked.
- If  $n$  is the largest possible integer, then the loops 1 and 3 may be infinite because the condition `|i <= n|` is going to be true for all possible `|i|`.

Questions?

## 5. Loops

---

## for $\rightarrow$ 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  
}
```

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

## do-while → for

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

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

```
// SOLUTION
```

```
BODY
```

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



Questions?

## 6. Calculating Sums

---

# From Series to Loop

Mathematical sums can be turned into loops

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

Becomes

```
int n = 0;
int sum = 0;

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

# From Series to Loop

## Taylor Series on **code** expert

Write a program that calculates  $\sin(x)$  up to six decimal places

Hint: What loop should be used here?

Use the MacLaurin Series.

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

## Task

- Try with pen and paper (10min)
- Try implementing together with person next to you in **code** expert (10min)

Questions?

## 7. Tips for **code** expert

---

# Tips for **code expert**

## **Tasks 1 and 2: “Loop mix-up”**

- If you can't figure out the loops right away, try plugging in a few numbers

## **Task 3: “Loop Analysis”**

- Q2: What values can variables of type `unsigned int` take?

## 8. Outro

---



# General Questions?

Till next time!

Cheers!