

CH-231-A

**Algorithms and Data Structures**

ADS

**Lecture 1**

Dr. Kinga Lipskoch

Spring 2022

# Who am I?

- ▶ PhD in Computer Science at the “Carl von Ossietzky” University of Oldenburg
- ▶ University lecturer at the Computer Science Department
- ▶ Joined Jacobs University in January 2013
- ▶ Office: Research I, Room 94
- ▶ Telephone: +49 421 200-3148
- ▶ E-Mail: [k.lipskoch@jacobs-university.de](mailto:k.lipskoch@jacobs-university.de)
- ▶ Office hours: Mondays 10:00 - 12:00  
[Click to join Teams meeting](#)

# Agenda Today

- ▶ Introduction
  - ▶ Syllabus and Organization
  - ▶ Goals
- ▶ More C++ programming related to data structures

# Online Resources

- ▶ Course website  
[https://grader.eecs.jacobs-university.de/courses/ch\\_231\\_a/2022\\_1/](https://grader.eecs.jacobs-university.de/courses/ch_231_a/2022_1/)
- ▶ Slides and homework will be uploaded there
- ▶ Use Grader for homework submission (change semester to Spring 2022)

## Teaching Assistants and Grading Criteria

- ▶ Adane, Tewodros Admassu
- ▶ Agrawal, Mahiem
- ▶ Banjade, Abiral
- ▶ Chowdhury, Priontu
- ▶ Kim, Eunhong
- ▶ Tafa, Bering
- ▶ Tasellari, Flavia
- ▶ Thapa, Prashiddha Dhoj
- ▶ Wong, Yen Ling
- ▶ Submit ZIP file containing one PDF file and source code files with makefile
- ▶ Grading criteria

[https://grader.eecs.jacobs-university.de/courses/ch\\_231\\_a/2022\\_1/Grading\\_Criteria\\_ADS.pdf](https://grader.eecs.jacobs-university.de/courses/ch_231_a/2022_1/Grading_Criteria_ADS.pdf)

## Grader not Publicly Visible

- ▶ You can access Grader from campus without any additional connection or software
- ▶ To access Grader from outside of campus you need to use a VPN (Virtual Private Network) connection
- ▶ Tutorials from the Jacobs IRC IT team on how to install a VPN client:

<https://teamwork.jacobs-university.de/display/ircit/VPN+Access>

## Missing Homework, Quizzes, Exams according to AP

- ▶ [https://www.jacobs-university.de/sites/default/files/bachelor\\_policies\\_v4.pdf](https://www.jacobs-university.de/sites/default/files/bachelor_policies_v4.pdf) (page 15 - 16)
- ▶ Illness must be documented with a sick certificate
- ▶ Sick certificates and documentation for personal emergencies must be submitted to the Student Records Office by the third calendar day
- ▶ Predated or backdated sick certificates will be accepted only when the visit to the physician precedes or follows the period of illness by no more than one calendar day
- ▶ Students must inform the Instructor of Record before the beginning of the examination or class/lab session that they will not be able to attend
- ▶ The day after the excuse ends, students must contact the Instructor of Record in order to clarify the make-up procedure
- ▶ Make-up examinations have to be taken and incomplete coursework has to be submitted by no later than the deadline for submitting incomplete coursework as published in the Academic Calendar

# Content

- ▶ This course introduces a basic set of data structures and algorithms that form the basis of almost all computer programs
- ▶ The data structures and algorithms are analyzed in respect to their computational complexity with techniques such as worst case and amortized analysis = method for analyzing a given algorithm's complexity, or how much of a resource, especially time or memory, it takes to execute
- ▶ Topics: fundamental data structures (lists, stacks, trees, hash tables), fundamental algorithms (sorting, searching, graph traversal)



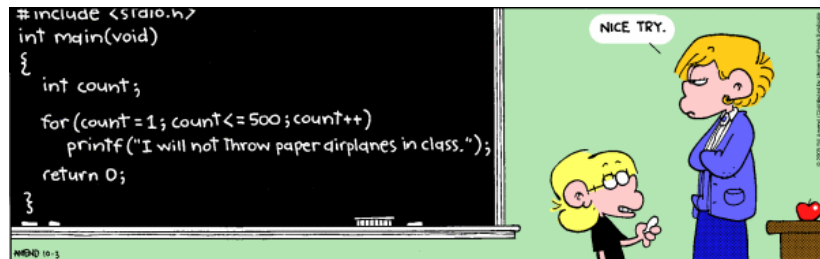
# Objectives

Learn about:

- ▶ Fundamental algorithms for solving problems efficiently
- ▶ Basic algorithmic concepts
- ▶ Analysis of algorithms
- ▶ Fundamental data structures for efficiently storing, accessing, and modifying data

# Requirements

Programming: freely choose between C or C++ or Python or Java if language is not enforced by the problem statement



# Lectures

- ▶ Time:
  - ▶ Tuesdays 8:15 – 11:00
  - ▶ Thursdays 11:15 – 12:30
- ▶ Location: Online via Teams for the first two weeks, later hopefully in person and hybrid

# Tutorials

- ▶ 2 weekly tutorials given by one TA
- ▶ Tutorial before homework deadline
- ▶ Online via Teams, Saturdays, 19:00 – 21:00
- ▶ Online via Teams, Sundays, 19:00 – 21:00

# Homework

- ▶ Homework
  - ▶ The homework assignments include theoretical and practical problems that tackle topics from the lectures
  - ▶ The homework assignments are handed out on a regular basis
- ▶ Submitting your homework
  - ▶ Extensions are possible only with an official excuse
  - ▶ Submit via Grader  
<https://grader.eecs.jacobs-university.de/>
- ▶ Homework deadline: Mondays, 23:00 sharp

# Final Exam

- ▶ ug 22
  - ▶ Module achievement:  $\geq 50\%$  average over all homework
  - ▶ Attendance of the final exam possible only if the module achievement is fulfilled
- ▶ ug 23 and ug 24
  - ▶ No module achievement
- ▶ Grading of the course: 100% final exam
- ▶ The final exam is a written exam
- ▶ Audit
  - ▶ Final exam cannot be written
  - ▶ Reach  $\geq 50\%$  in homework then audit will be granted, otherwise audit will not be granted

# Literature

- ▶ "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein, 3rd edition, MIT Press, 2009
- ▶ "The Art of Computer Programming", volumes 1-3 by Donald E. Knuth, Addison-Wesley, 3<sup>rd</sup> edition, 1997

# Syllabus of Course

- ▶ More C++ Programming
- ▶ Foundations
- ▶ Sorting and Searching
- ▶ Fundamental Data Structures
- ▶ Design Concepts
- ▶ Graph Algorithms
- ▶ Dynamic Programming
- ▶ Backtracking



# Templates

- ▶ Templates allow to write generic code, i.e., code which will work with different types
  - ▶ Again those types could be unknown at code time
- ▶ A template tells the compiler that “what is following” will deal with an unknown type
- ▶ Later a specific type will be provided and the compiler will substitute it and generate ad-hoc code

## Templates: Motivation

- ▶ Many times it is required to write different snippets of code which differ only in the types dealt with, but not in the underlying logic
  - ▶ Imagine the code to check for the existence of an element in an array of floats, or an array of pointers to a class, or an array of images
  - ▶ The logic is always the same
- ▶ So, why do not we write code which is parametric with respect to the possible types?

## Searching in a Vector

- ▶ Assuming that a comparison operator is defined, the following code captures the logic to locate an element in a vector

```
1 int seek(sometype A[], int n, sometype toseek) {  
2     for (int i = 0; i < n; i++)  
3         if (A[i] == toseek)  
4             return i;  
5     return -1;  
6 }
```

- ▶ Should write different versions if sometype is `int`, or `float`, or `Complex`, or ...?

# Templates: Functions and Classes

Type parameterization can be introduced for:

- ▶ **Functions:** like in the previous example; this helps in developing “algorithms”; you can concentrate on the logic, rather than on type details
  - ▶ Also, this decreases your coding time
- ▶ **Classes:** helps in developing “generic” classes; think about an array: the underlying logic is the same, whether it holds elements of type `int`, `Car`, `Student`, `double`, etc.
  - ▶ Again: concentrate on developing a working generic version

## Templates: Basic Syntax (1)

- ▶ Two keyphrases are involved: `template class` and `template typename`
- ▶ They are functionally equivalent
- ▶ Template function: `template_function.cpp`

```
1 template <class T>
2 class Something {
3     T *p;
4     public: Something() { p = new T[100]; }
5 };
```

- ▶ Here the type `T` is not known, it will (and must) be specified when declaring instances of the class `Something`

## Templates: Basic Syntax (2)

- ▶ When declaring an instance, the type is provided between angular brackets

```
1 int main(int argc, char** argv) {  
2     Something<int> ints;  
3     Something<char*> chars;  
4     Something<student> studentsome;  
5 }
```

- ▶ The compiler will generate the code necessary for the three different types
- ▶ `templatesone.cpp`