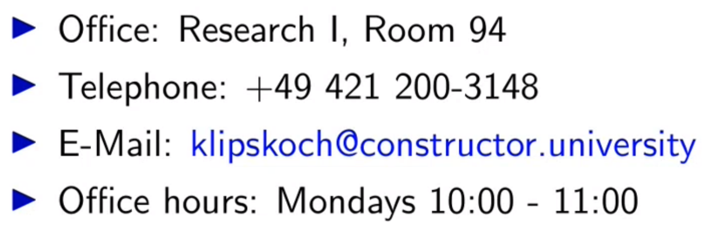
Algorithms and Data Structures

Author: Daria Shutina

Organization stuff

Professor Kinga Lipskoch



- Office: Research I, Room 94
- Telephone: +49 421 200-3148
- E-Mail: klipskoch@constructor.university
- Office hours: Mondays 10:00 - 11:00

Lecture slides: https://grader.eecs.jacobs-university.de/courses/ch_231_a/2023_1/

Homeworks:

Use Grader for homework submission. Do not forget to change semester from Fall 2022 to Spring 2023.

Submit ZIP file containing one PDF file and source code fileswith makefile.

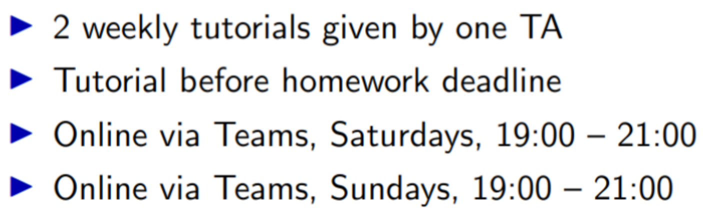
Homeworks are not mandatory.

VPN for remote access to Jacobs Net:

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

Domain name: vpnasa.jacobs-university.de (Jacobs VPN)

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

23-02-02

Templates

Templates allow to write generic code, i.e., code which will work with different types. Later a specific type will be provided and the compiler will substitute it.

Motivation for using templates is to eliminate snippets of code that differ only in the type and do not influence on the logic.

While using templates, operators for some types need to be overloaded.

Basic syntax

Type parameterization in classes:

```cpp
template <class T>
class MyClass {
    T i;
public:
    MyClass() = default;
    MyClass(T arg) : i(arg) {}
};

int main() {
    MyClass<int> aboba;
}
```

Type parameterization in functions:

```cpp
template <class T>
void my_function(T arr[], int size) {
    for (int i = 0; i < size; ++i) {
        std::cout << arr[i] << ' ';
    }
}
```

Note: `class` in `<class T>` can be replaced with `typename`.

Overloading an operator

```cpp
template <class T>
class MyClass {
    T array[size];
public:
    BoundedArray(){};
    T& operator[](int);
};

...

template<class T>
T& MyClass<T>::operator[](int pos) {
    if ((pos < 0) || (pos >= size))
        exit(1);
    return array[pos];
}
```