Mathematical Software Programming (02635)

Module 13 — Fall 2016

Instructor: Martin S. Andersen

About the exam

Where

101/Hal 2

When

December 7, 2016 from 9:00-13:00

Format

- Written exam, individual and all digital
  - Connect to wireless network **EKSAMEN**
  - Go to `http://onlineeksamen.dtu.dk`
- One document with multiple choice and programming questions
  - Submit your answers as a PDF file: `http://onlineeksamen.dtu.dk`
  - Include your code in the document or as an attachment (ZIP file)

More information is available here: Digital eksamen / Digital Examination

This week

Topics

- Introduction to object-oriented programming and C++

Learning objectives

- Describe and use basic object-oriented programming concepts such as classes and objects
- Analyze the runtime behaviour and the time and space complexity of simple programs

Templates

Generic programming via function templates and class templates

Example: max function

```
#include <iostream>

template <class T>
const T& max (const T& a, const T& b) {
  return (a<b)?b:a;
}

int main(void) {

    std::cout << max(1.0,2.0) << std::endl;
    std::cout << max(5,-3) << std::endl;
    std::cout << max('a','z') << std::endl;

    return 0
}
```

The standard template library (STL)

```
// using the vector class template (requires <vector> header)
std::vector<double> v;
v.push_back(1.0);          // append 1.0 to back
v.insert(v.begin(),2.0);   // append 2.0 to front
std::cout << v[0] << "\n"
          << v[1] << "\n"
          << v.size() << "/" << v.capacity() << "\n";
```

using v.at( ) in order to have boundary check, but it comes with a cost

```
// using the list class template (requires <list> header)
std::list<int> l;
l.push_back(2);            // append 2 to back
l.push_front(4);           // append 4 to front
std::list<int>::iterator it;  // declare list "iterator"
for (it=l.begin(); it!=l.end(); it++)
   std::cout << *it << "\n";
```

What about complexity? Should I use a `list` or a `vector`?

complexity with capacity reallocation: O(log n)

Reading numbers from a file

```
#include <iostream>
#include <ifstream>

int main(void) {

    double val;

    std::ifstream myfile;
    myfile.open("myfile.txt");
    if (myfile.fail()) {
        std::cerr << "Error: " << strerror(errno) << std::endl;
        std::exit(-1);
    }
    while (myfile >> val) {
        // do something
    }

    myfile.close();

}
```

Review and questions