Mathematical Software Programming (02635)

Lecture 13 — December 6, 2018

Instructor: Martin S. Andersen

Fall 2018

About the exam

When

December 11, 2018 from 9:00-13:00 (special needs: 9:00-14:00)

Where

eksamensplan.dtu.dk

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 (two versions: Eng./Da.)
  - Submit your answers as a PDF file: http://onlineeksamen.dtu.dk
- One ZIP file with templates for questions that require some code
  - Attach/upload your code (e.g. as a ZIP file): http://onlineeksamen.dtu.dk
- You always have the right to hand in *handwritten* answers

More information is available here: Digital eksamen / Digital Examination

This week

Topics

- Introduction to object-oriented programming and C++
- C/C++ API and scripting languages
- Review and questions

Learning objectives

- Describe and use basic object-oriented programming concepts such as classes and objects
- Analyze the run-time behavior 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)?a:b;
}

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 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`?

The standard template library (STL)

vector is implemented as a dynamic array

- contiguous storage allows fast random access
- fast insertion/deletion at the end of the array
- insertion/deletion at the end: `pop_back()` and `push_back()`
- insertion/deletion at any position: `insert()` and `erase()`

list is implemented as a doubly-linked list

- slow random access
- fast insertion/deletion in any position
- insertion/deletion at the front: `push_front()` and `pop_front()`
- insertion/deletion at the end: `push_back()` and `pop_back()`
- insertion/deletion at any position: `insert()` and `erase()`

Reading numbers from a text file

```cpp
#include <fstream>
#include <iostream>
#include <vector>
using namespace std;
int main(void) {
  double val;  vector<double> v;
  fstream myfile;
  myfile.open("myfile.txt", ios::in);
  if (myfile.fail()) {
    cerr << "Error opening file.." << endl;
    exit(-1);
  }
  while (myfile >> val) v.push_back(val);
  myfile.close();
  cout << "Read " << v.size() << " numbers from file." << endl;
  return 0;
}
```

Application Programming Interface

- Specification that allows programs to communicate
- Extend MATLAB/Python/R/... with your functions written in C or C++

MATLAB example

- MATLAB API for other languages
- C MEX files

```
edit([matlabroot '/extern/examples/refbook/matrixDivide.c']);
```

Review and questions

- Trial exam (onlineeksamen.dtu.dk)
- Questions