# Advanced Software Development

Modus und Inhalte

# Termine

- 4 Blöcke zu je 6 Einheiten

- Di 20.10.2015, 8:30 – 13:30
- Di 3.11.2015, 8:30 – 13:30
- Di 1.12.2015, 8:30 – 13:30
- Di 15.12.2015, 8:30 – 13:30

# Modus

- Kombination aus Vorlesung und Übung
- Vorlesungsartiger Teil
  - Präsentation auf Folien
  - Einzelne Beispiele und Ergänzungen auf Tafel
- Übungsartiger Teil
  - Aufgaben zur Lösung als Hausübung

# E-Learning-Plattform (Moodle)

- Zugangsschlüssel: #ASD15-16#
- Folien als PDF verfügbar
- Übungsangaben
- Online-Übungsabgabe
- Online-Übungsbewertung

# Hausübungen

- Selbständige Ausarbeitung
- Beispiele mit unterschiedlichem Umfang
- Zeit: 2 Wochen
- Abgabe über Moodle
- Bewertung mittels Punkteskala
  - Punktezahl je nach Umfang
  - Bewertung z.T. stichprobenartig

# Beurteilung

- Gesamtnote ergibt sich aus
  - Übungen (40%)
  - Klausur (60%)
- Klausur
  - Allgemeine Verständnisfragen
  - Keine Detailfragen
  - Keine vollständigen Programmieraufgaben

# Benötigte Software

- C++ Compiler
  - Visual C++ 2015
  - gcc/g++ ab Version 4.9
  - Clang ab Version 3.4
- Zusätzliche Libraries für einzelne Aufgaben
  - Boost Library
  - TBB

# Geplante Themen

- Allgemeine Neuerungen in C++ 11/14
- Smart Pointer
- Funktionsobjekte
- Templates und Template-Metaprogrammierung
- Multithreading
- Parallele Programmierung
- Benutzerdefinierte Speicherverwaltung

# C++ 11/14

- Neue Versionen des C++ Standards
- Umfangreiche Spracherweiterungen
- Erweiterte Standardbibliothek
- Unterstützung durch aktuelle Compiler
  - C++ 11 in g++ seit Version 4.8 vollständig, C++ in Version 5
  - In Visual C++ 2015 noch nicht alle Elemente von C++ 11 und 14

# Smart Pointer

- Speicherverwaltung mit new/delete ist fehleranfällig
  - Memory Leaks
  - Dangling Pointers
- Smart Pointer ermöglichen u.a.
  - Automatische Freigabe zu definiertem Zeitpunkt (ohne explizites delete)
  - Sichere gemeinsame Verwendung von Objekten

# Templates

- Template-Spezialisierung
  - Spezielle Varianten für bestimmte Parameter-Typen
  - Partielle Spezialisierung
- Variadic Templates
- Template-Metaprogrammierung

# Funktionsobjekte

- Call Wrapper
  - Verallgemeinerung der STL Binder (*bind1st* und *bind2nd*)
  - Mit beliebigen Funktionen und Methoden verwendbar
- Lambda Expressions
  - Ausdrücke als unbenannte Funktionsobjekte
  - Neuerung in C++ 11

# Parallele Programmierung

- C++ Threads
  - neu in C++ 11
- Intel TBB, Microsoft PPL
  - Plattformunabhängige Parallelisierungs-Library
  - Parallelisierung auf höherer Ebene
  - Task-basiert statt thread-basiert
  - Parallele Algorithmen (z.B. *parallel_sort*)

# Benutzerdefinierte Speicherverwaltung

- Standard new/delete müssen alle möglichen Szenarien berücksichtigen
- Performance-Gewinn durch benutzerdefinierte Speicherverwaltung
  - Spezielle Allokatoren für bestimmte Klassen
  - Memory Pools
- Insbesondere für Skalierbarkeit von parallelen Anwendungen ein Thema