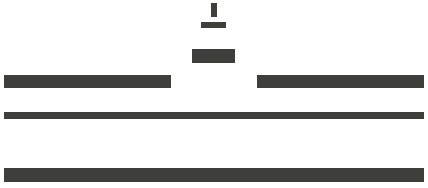

Westfälische
Wilhelms-Universität
Münster

Übung zur Vorlesung Wissenschaftliches Rechnen — Wintersemester 2019/20

Mini Überblick zu C++

living.knowledge
WWU Münster

Marcel Koch

16. Oktober 2019

Organisatorisches

- Ausgabe der Übungszettel jeweils am Donnerstag

Organisatorisches

- Ausgabe der Übungszettel jeweils am Donnerstag

- Abgabe Montags, pünktlich 10:00 Uhr, Briefkasten 112
- Abgabe in festen Zweiergruppen (jetzt Partnersuche)

Organisatorisches

- Ausgabe der Übungszettel jeweils am Donnerstag
- Abgabe Montags, pünktlich 10:00 Uhr, Briefkasten 112
- Abgabe in festen Zweiergruppen (jetzt Partnersuche)
- Schriftliche Abgabe, Code zusätzlich per e-mail
- Code muss kompilieren, sonst wird er nicht bewertet
- Jeder sollte sich am programmieren beteiligen, nur so lernt man was

Organisatorisches

- Ausgabe der Übungszettel jeweils am Donnerstag

- Abgabe Montags, pünktlich 10:00 Uhr, Briefkasten 112
- Abgabe in festen Zweiergruppen (jetzt Partnersuche)

- Schriftliche Abgabe, Code zusätzlich per e-mail
- Code muss kompilieren, sonst wird er nicht bewertet
- Jeder sollte sich am programmieren beteiligen, nur so lernt man was

- Interesse an mattermost?
- e-mail an `marcel.koch@uni-muenster.de`

Eingliederung C++

Eingliederung C++

- Multiparadigmen Sprache
  - objektorientiert, generisch, funktional, ...

Eingliederung C++

- Multiparadigmen Sprache
  - objektorientiert, generisch, funktional, ...
- C++ Code muss kompiliert werden

Eingliederung C++

- Multiparadigmen Sprache
  - objektorientiert, generisch, funktional, ...
- C++ Code muss kompiliert werden
- Hardwarenah und hohes Abstraktionsniveau

Eingliederung C++

- Multiparadigmen Sprache
  - objektorientiert, generisch, funktional, ...
- C++ Code muss kompiliert werden
- Hardwarenah und hohes Abstraktionsniveau
- Stark typisiert

Eingliederung C++

- Multiparadigmen Sprache
  - objektorientiert, generisch, funktional, ...
- C++ Code muss kompiliert werden
- Hardwarenah und hohes Abstraktionsniveau
- Stark typisiert
- Manuelle Ressourcenverwaltung

Objektorientiert

```cpp
class ExplicitEuler1D{
  using F = std::function<double(double)>;
public:
  ExplicitEuler(F f_) : f(f_) {}

  void apply(double t, double dt, double y_old, double& y_old){
    // compute explicit euler step
  }
private:
  F f;
}
```

Generisch

```
template<typename Vector>
class ExplicitEuler{
  using F = std::function<Vector(Vector)>;
public:
  ExplicitEuler(F f_) : f(f_) {}

  void apply(double t, double dt, Vector& y_old, Vector& y_old){
    // compute explicit euler step
  }
private:
  F f;
}

using Vector = std::array<double, 2>;
```

Funktional

```cpp
using Vector = std::array<double,1>;

double c = 0.1;
auto f = [](Vector& v) { return c * v;};

ExplicitEuler method(f);
```

```cpp
using Vector2D = std::array<double, 2>;
std::vector<Vector2D> data;
/* initialize data */
std::sort(begin(data), end(data),
    [] (const Vector2D& a, const Vector2D& b)
    { return a[0] < b[0] || (a[0] == b[0] && a[1] < b[1]);});
```

Standard Template Library

Containers

- Array
- Vector
- (unordered_)Map/Set
- List

Standard Template Library

Containers

- Array
- Vector
- (unordered_)Map/Set
- List

Utilities

- Pair/Tuple
- function
- smart pointer (shared/unique_ptr)

Standard Template Library

Algorithms

- find
- all_of
- max
- copy

Standard Template Library

Algorithms

- find
- all_of
- max
- copy
- sort
- rotate
- transform
- accumulate

C++ 11

- `auto` Deklaration

C++ 11

- `auto` Deklaration
- Range based for loop `for(auto r: rechtecke)`

C++ 11

- `auto` Deklaration
- Range based for loop `for(auto r: rechtecke)`
- lambda functions

C++ 11

- `auto` Deklaration
- Range based for loop `for(auto r: rechtecke)`
- lambda functions
- constructor delegation

C++ 11

- `auto` Deklaration
- Range based for loop `for(auto r: rechtecke)`
- lambda functions
- constructor delegation
- `nullptr`

C++ 11

- `auto` Deklaration
- Range based for loop `for(auto r: rechtecke)`
- lambda functions
- constructor delegation
- `nullptr`
- Move constructor `Class(Class&& o)`

Ressourcen

- `http://www.cppreference.com`
- C++ core guidelines
- `http://www.isocpp.com`
- Boost C++ Libraries

Was braucht man?

- Texteditor
  - Vim, Emacs, Sublime, Atom, VSCode, XCode, ...

Was braucht man?

- Texteditor
  - Vim, Emacs, Sublime, Atom, VSCode, XCode, ...
- Kompiler
  - g++, clang, icc(, msvc)

Was braucht man?

- Texteditor
  - Vim, Emacs, Sublime, Atom, VSCode, XCode, ...
- Kompiler
  - g++, clang, icc(, msvc)
- Buildsystem?
  - Make, CMake, ...

Was braucht man?

- Texteditor
  - Vim, Emacs, Sublime, Atom, VSCode, XCode, ...
- Kompiler
  - g++, clang, icc(, msvc)
- Buildsystem?
  - Make, CMake, ...
- alles in einem: IDE
  - CLion, Eclipse, Code::Blocks, QtCreator(, Visual Studio)

Getting started

- Laden Sie das Programm „Hallo Welt“ von der Vorlesungshomepage
- Öffnen Sie es (Texteditor/Entwicklungsumgebung Ihrer Wahl)
- Kompilieren Sie es, beispielsweise durch
  ```
  g++ -o hallowelt hallowelt.cc
  ```
- Führen Sie das Programm aus

- Bearbeiten Sie nun die Anwesenheitsaufgaben