Fachhochschule Nordwestschweiz
Hochschule für Technik

Programmieren in C++

Christian Lang (Lac)

20. September 2019

Einführung

Inhalt

- Über mich
- Übersicht über das Modul
- Literatur & Links
- Geschichte von C++
- Einsatz von C/C++ heute
- Modernes C++
- Compiler Support

Über mich

Christian Lang

- Jahrgang 1986
- gelernter Elektroniker
- Studium: Bachelor und Master (MSE) an FHNW
- Arbeitgeber seit 2014: Supercomputing Systems in Zürich
  - High-Performance Abteilung
- Spezialgebiete:
  - Embedded Plattformen
  - Linux
  - C++
  - Optimierung
- Lernbeauftragter FHNW seit 2017: `chp` und `prcpp`

Modul-Übersicht

- Leitidee
  - Ausrichtung am neusten Standard (Modern C++) C++11/14/17
  - Unterschiede und Gemeinsamkeiten mit Java aufzeigen
  - Praktischer Einsatz von C++ erleben
- Ablauf
  - 2 Stunden Vorlesung (Folien) / 1 Stunde Übung
  - Arbeitsblätter pro Foliensatz (selbständig bearbeiten)
  - Code Dojos: z.B: codewars.com
  - 3 Testatübungen (teilweise optionale Teilaufgaben)
  - 2 Klausuren (2/4 A4-Seiten Zusammenfassung, kein Laptop etc.)
- Erforderliche Vorkenntnisse
  - OO-Programmierung in Java
  - System-Programmierung: C

Literatur

- Der C++ Programmierer. C++ lernen - professionell anwenden - Lösungen nutzen. Ulrich Breymann. Hanser, 2015.
- Schrödinger programmiert C++: Jetzt mit C++14. Dieter Bär. Rheinwerk, 2015.
- Einführung in C++. Torsten Will. Galileo Computing, 2014.
- C++11 programmieren: 60 Techniken für guten C++11-Code. Torsten Will. Galileo Computing, 2012.
- Einführung in die Programmierung mit C++. B. Stroustrup. Pearson, 2010.
- Effective Modern C++. Scott Meyers. O'Reilly, 2014.

Links

Standard Library etc.

- cplusplus.com
- cppreference.com

Features & Style

- Overview Modern C++ Features
- Cpp Core Guidelines
- Google C++ Style Guide

Online-Tools

- godbolt.org
- cppinsights.io
- wandbox.org
- quick-bench.com

Geschichte C

- 1970
  - Dennis Ritchie entwickelt C für Unix in den AT&T Bell Labs
- 1978
  - Brian Kerningham und Dennis Ritchie publizieren C
- Standardisierung von C
  - 1989: ANSI C89
  - 1990: ISO C90 (ISO/IEC 9899:1990)
- Erweiterungen
  - 1999: ISO C99 (ISO/IEC 9899:1999)
  - 2011: ISO C11 (ISO/IEC 9899:2011)

Geschichte C++

- 1979
  - “C with Classes” wird von Bjarne Stroustrup in den AT&T Bell Labs entwickelt
  - “C with Classes” erweitert C um ein Klassenkonzept, welches sich an Simula-67 anlehnt
- 1983
  - Umbenennung in C++
  - viele Erweiterungen
    - virtuelle Funktionen, Überladen von Funktionsnamen und Operatoren, Referenzen, Konstanten, änderbare Speicherverwaltung und eine verbesserte Typüberprüfung
- 1985
  - erste C++ Referenzversion (noch nicht standardisiert)

Geschichte C++

- 1989
  - zweite Version (noch nicht standardisiert)
  - Erweiterungen
    - Mehrfachvererbung, abstrakte Klassen, statische Elementfunktionen, konstante Elementfunktionen und die Erweiterung des Schutzmodells um protected
- 1990-1998
  - Erweiterungen
    - Templates, Ausnahmen, Namensräume, neuartige Typumwandlungen und boolesche Typen
  - Erweiterung der C-Standardbibliothek um Datenströme und um die Standard Template Library (STL)
  - Standardisierungsverfahren
- 1998
  - Endgültigen Normierung (ISO/IEC 14882:1998)

Geschichte C++

- 2003
  - Nachbesserung der Norm von 1998 (ISO/IEC 14882:2003)
- 2005
  - Technical Report 1 (TR1): Erweiterungen, die 2003 nicht standardisiert wurden
- 2011
  - Einführung des neuen C++-Standards C++11
  - Modern C++
  - Ab jetzt werden im 3-Jahres-Rythmus neue Features und Verbesserungen in den Standard aufgenommen
  - Dies umfasst jeweils Sprach- wie auch Library-Features

Einsatz von C/C++ heute

- C war die erste High-Level-Sprache
  - weit verbreitet: z.B. im Linux-Kernel oder Treiber-Entwicklung
- C++ als Nachfolger mit OOP u.a. Paradigmen
  - grosse Userbase
  - Rückwärtskompatibel (erlaubt einfaches Einbinden von C-Libraries)
  - Standartisiert (auch 20 Jahre altes Programm kompiliert noch)
  - erlaubt Speicheroptimierung (Programm- und Arbeitsspeicher)
  - erlaubt high- und low-level Laufzeit-Optimierung
  - gut geeignet für spezielle Plattformen (Embedded, Mikroprozessoren, etc.)

Modern C++ Features: C++11

- Move semantic
- Smart pointers
- Threading & Asynchronität
- Lambda expressions
- Initializer lists
- Range based for loops
- Variadic templates
- Improved type deduction: `auto`, `decltype`
- Compile time: `static_assert`, `constexpr`

Modern C++ Features: C++14

- Generic lambda expressions
- User defined literals
- return type deduction
- `std::make_unique`

Modern C++ Features: C++17

- Parallel algorithms
- Folding expressions
- Improved type deduction
- Structured bindings
- Filesystem abstraction: `std::filesystem`
- `std::variant`, `std::optional`
- `std::string_view`

Modern C++ Features: C++20

- Modules
- Contracts
- Concepts
- Coroutines
- Ranges-Library
- Typesafe Format: Adaption von libfmt

Compiler Support

Feature	GCC	clang	MSVC
C++11	since 5	since 3.8	since 19.0
C++14	since 5	since 3.4	since 19.10
C++17	most since 8	most since 4	since 19.15
C++2a	many since 9	many since 8	some since 19.20

Link

cppreference - C++ compiler support