Modernes C++

...für Programmierer

Unit 00a: Einleitung

by

Dr. Günter Kolousek

1. September 2018

Wie alles begann

1967 Simula-67

- **erste** OO Sprache & Koroutinen!

1972 C

- Dennis Ritchie

1978 *The C Programming Language*

- Brian Kernighan & Dennis Ritchie $\rightarrow$ K&R C

1979 C with Classes

- Bjarne Stroustrup
- Simula-67 & C

1984 C++

1989 C++ 2.0

1990 *Annotated Reference Manual* (”ARM”)

Versionen

C++98 erste ISO Standardisierung von C++!

C++03 Fehlerbereinigung & Standardbibliothek

TR1 Library Technical Report 1 (2005)

C++11 Evolutionsschritt!

- einfacher! (Lehren, Lernen, Verwenden)
- rückwärtskompatibel
- bessere Performance
- Speichermodell!
- bessere Anbindung an moderne Hardware
- Verbesserung der Standardbibliothek
  - Threads und thread-lokale Daten
  - atomare Typen
  - Mutex und Lock, Bedingungsvariable
  - Tasks

Versionen – 2

C++14 Fehlerbereinigung & einige neue Features

- Rückgabetypableitung von Funktionen
- Verbesserungen bei Lambdaausdrücke
- Binäre Literale
- benutzerdefinierte Standardliterale, wie z.B. `"abc"s` oder `3h`
- Reader-Writer Locks

C++17 Viele neue Features!

- template deduction
- parallele Algorithmen, `std::filesystem`,
- `optional`, `variant`, `any`, `string_view`, `shared_mutex`, `scoped_lock`
- structured bindings
- guaranteed copy elision
- inline variables
- Variablendefinition in `if`, `switch`

Versionen – 3

C++20 erwarteter Evolutionsschritt!

- ”sicher”
  - Concepts
  - weitere Verbesserung der Lambdaausdrücke
- wahrscheinlich/hoffentlich:
  - atomare Smart Pointer
  - `std::future` Erweiterungen
  - Latches und Barriers
  - Coroutinen
  - Task-Blöcke
  - Ranges
  - Modulekonzept
  - Netzwerkprogrammierung basierend auf asio

Charakterisierung 1

gemäß Stroupstrup:

- C++ ist…
  - allgemein verwendbare Programmiersprache
  - zum Entwickeln und Verwenden eleganter Abstraktionen
- grundlegende Entwurfsziele
  - nahe an der Hardware
  - hoch performante Programme

Charakterisierung 2

- Paradigmen
  - prozedural, objektorientiert
  - funktional
  - generisch
- Typisierung
  - statisch
  - streng
- Ressourcenverwaltung: manuell oder RAII
- stabil: hohe Kompatibilität zu früheren Versionen
- unabhänging von einem Unternehmen
- weit verbreitet!

Anwendungsgebiete – Einsatz

- eingebettete Systeme jeder Größe
  - Uhren, Handys, Router, DVD Player,…
  - Waschmaschinen, Kühlschränke,…
  - Medizintechnik, Automotive, Flugzeug, Raumfahrt, Prozesssteuerungen,…
- Hosts jeder Größe
  - Laptop, PC
  - Server
  - Hochleistungsrechner

Anwendungsgebiete – Art

- Systemprogrammierung
  - Treiber, Betriebssysteme, Compiler, Datenbanken, Netzwerkstacks
  - eingebette Systeme
- Anwendungsprogrammierung
  - Büroanwendungen, Grafikprogramme, Spiele
  - Banksektor und Echtzeitverarbeitung in Finanzmärkten
  - Verarbeitung von Multimediadaten (Bilder, Audio, Video)
  - Wissenschaftliche und numerische Anwendungen: Simulationen (z.B. Crashtests, Statik, Metereologie,…), Medizin, Bioinformatik, Genetik, Physik,…
  - Telekom
  - Serverprogrammierung
  - Deep learning (neuronale Netze)

Firmen, Einsatz & Anwendungen

- Google: Chrome, Chrome OS, google.com, Google File System, youtube (Python & C++), TensorFlow (deep learning, Python & C++)
- Microsoft: Windows, MS Office, MS SQL Server, IE, Edge, IIS, Outlook, MS Exchange Server, Visual Studio, Visual C#/C++, Visual Basic, .Net CLR, DirectX, MS Media Player, CNTK (deep learning)
- Apple: Mac OSX (Teile), Safari, iPod UI
- Amazon: amazon.com
- Mozilla: Firefox
- Oracle: Oracle DB, MySQL, JVM
- Adobe: Photoshop, Acrobat Reader/Distiller, InDesign,…
  - Photoshop ~4.5 Mio LoC!
- Facebook: Facebook

Firmen, Einsatz & Anwendungen – 2

- Bloomberg: ”global business and financial information and news leader”
- CERN: ROOT (scientific SW framework),…
  - Teilchenbeschleuniger Large Hadron Collider (26km)
- NASA: JPL (Mars Roboter),…
- UBIMET (international, Sitz Wien, Wetter…)
- verschiedene Anwendungen
  - KDE
  - LibreOffice, Evernote, VLC
  - MariaDB, MongoDB, ScyllaDB, Couchbase Server, LevelDB, RocksDB
  - Opera
  - Paypal
  - gcc, clang
  - deep learning: caffe, OpenNN,…

Warnung

- C++ ist eine komplexe Sprache!!!
  - nur das verwenden, das man beherrscht!
  - $\rightarrow$ Modern C++
- bestimmte Sprachefeatures von C++ sind
  - nicht definiert
  - implementierungsabhängig
- Compiler optimieren **sehr**!
- Compiler dürfen mehr ”erledigen”
  - solange es nicht dem Standard widerspricht
- auch Compiler haben Fehler…
- daher: Testen, testen, testen!

Zweite (und letzte) Warnung

> *C makes it easy to shoot yourself in the foot;*
> *C++ makes it harder, but when you do it blows your whole leg off*
>
> *– Bjarne Stroustrup*

Und hier sind Guidelines, die helfen (falls man diese befolgt…), aber Achtung: ausgedruckt wären dies ~540 Seiten!

Quellen

- http://cppreference.com
- The C++ Programming Language, 4th ed., Bjarne Stroustrup
- The C++ Standard Library, 2nd ed., Nicolai M. Josuttis
- C++ Concurrency in Action, Antony Williams
- C++11 programmieren, Torsten T. Will
- Effektives modernes C++, Scott Meyers
- API Design for C++, Martin Reddy
- C++ Recipes, Bruce Sutherland

Quellen – 2

- Discovering Modern C++, Peter Gottschling
- Der C++ Programmierer, 2. Auflage, Urlich Breymann
  - aktuell: 4. Auflage!
- C++ Das umfassende Handbuch, 3. Auflage, Jürgen Wolf
- C++ Standard Library Quick Reference, Peter Van Weert & Marc Gregoire
- …