C++ Stil-Richtlinien (Seminar Fortgeschrittene Programmierkonzepte) SS 2017

Prof. Dr. Hans-Jürgen Buhl

Praktische Informatik, Fachgruppe Mathematik und Informatik
Fakultät für Mathematik und Naturwissenschaften
Bergische Universität Wuppertal

$\mathcal{P}_I$

24. April 2017

Themenschwerpunkte

Die Berücksichtigung guten Stils, das heißt die Einhaltung bewährter, sich als qualitätssteigernd erwiesener objektorientierter Praktiken und Sprechweisen kann den Wert von Softwareprodukten außerordentlich steigern.

Es geht in diesem Seminar um die geeignete Namensgebung, die Sprachauswahl, den Einsatz von objektorientierten Design-Patterns, Debugging-unterstützende Kommentare, ...

Themenschwerpunkte

In den Seminarvorträgen sollen die Vortragenden übliche qualitätssteigernde Praktiken[2], [2] vorstellen und auch anhand eigener Programmbeispiele illustrieren:

- "Don't patch bad code – rewrite it."
- "Test input for plausability and validity."
- "Let the data structure the program."
- "Use type safe enums" (C++11 enum classes, scoped enums)
- "Modularize. Use procedures and functions." (... and C++20(?) modules (N 4637))
- "Use type safe unions" (C++17 std::variant)
- "Use sequence containers" (Improving type safety and security with sequence containers), ...

Organisatorisches: Zum Seminar

- **Termin:** Mo 12-14, Raum D.13.15
- **Zielgruppe:** Bachelor-Studierende
- **Voraussetzungen:** Kenntnisse der Objektorientierten Programmierung, C++11 (OOP-Vorlesung)
- **Teilnahme:** Verbindliche Anmeldung bei Vorbesprechung
- **Zu erbringende Leistung:** Jede(r) Teilnehmer(in) arbeitet sich in das gewählte Thema ein,
  hält einen (Beamer-)Seminarvortrag mit Illustrationen durch Programmbeispiele zum gewählten Stil-Themenbereich.
- **Workload:** 90 Stunden! (laut Modulbeschreibungen)

Organisatorisches: Terminplanung

- **Vorbesprechung mit Themenvergabe:**
  Montag, den 24. April 2017, 12 Uhr c.t. in Raum D.13.15
- danach Zeitraum zur Einarbeitung ins Thema, zur Literaturrecherche, zur Vorbereitung des Vortrags und Rücksprache mit dem Betreuer, für Probevorträge im Bekanntenkreis, für Korrekturlesen, ...
- **mögliche Vortragstermine:**
  12. Juni, 19. Juni, 26. Juni,
  3. Juli, 10. Juli, 17. Juli, 24. Juli

Je nach Studiengang bzw. Fachrichtung können folgende **Nachweise** erworben werden:

- **Bachelor-IT (PO 2006/2009):**
  BIT 15 (Pflichtmodul Ergänzende Wissenschaften, 1 SWS, 3 LP)
- **Bachelor-IT (PO 2011):**
  FBE0071 Erg. Wissenschaften, Techniken des wiss. Arbeitens, 2 SWS, 3 LP)
- **Bachelor Applied Science (PO 2007):**
  Seminar zur Informatik (2 SWS, 3 LP)
- **Komb. Bachelor of Arts, Informatik:**
  Seminar zur Informatik (2 SWS, 3 LP)
- **Alle Studiengänge (freiwillige Teilnahme):**
  allg. Leistungsnachweis „Seminar“ (Schein)

Organisatorisches: erfolgreiche Teilnahme

- Auswahl Thema, Einarbeitung und Beschäftigung mit diesem Thema
- Konkret: Selbständige Literaturrecherche,
  didaktische Aufbereitung des Themas,
  Implementierung eigener und vorgegebener Beispiele
- Ausarbeitung Vortrag (max. 40 Min. + 5 Min. Diskussion)
- Schriftliche Ausarbeitung des Themas (6-8 Seiten)
  (mit Titelseite und vollständigen Quellenangaben, eventuell Index)
- Mindestens ein Termin beim Betreuer vor dem Vortrag
  (Deadline: Gliederung+Aufbau 2 Wochen vorher;
  Vortragsfolien oder -Präsentation 1 Woche vorher)

Organisatorisches: erfolgreiche Teilnahme (Forts.)

- Bereitstellung von Ausarbeitung und Beispielprogrammen auf Web-Server der Fachgruppe Mathematik
- Rechtzeitig vor Beginn des Vortrags erscheinen (mind. 15 Min vorher), rechtzeitiger Test/Kontrolle der techn. Ausstattung (Rechner, Beamer, Laserpointer, Tafel, Kreide, ...)
- Anwesenheitspflicht bei allen Vorträgen
- Aktive Mitarbeit wird erwartet (Fragen, Diskussion, ...)

Themenvorschläge

1. Klassische Programmier-Stilregeln
   "Don't just echo the code with comments – make every comment count."
   "Löse im Fehlerfall eine Exception aus." ...
2. Stil-Sentenzen/Maximen für C++(11/14/17/20(?))
   "Distinguish between () and {} when creating objects."
   "Prefer alias declarations to typedefs." ...
3. Nutze Laufzeit-Zusicherungen zur Validierung von Code
   "assert (myInt != nullptr);"
   "assert(size <= LIMIT);" ...
4. Namenswahl/Sprachauswahl
   Im OOP sind Klassennamen Substantive, die in CamelCase gegliedert sind.
   Namen müssen lesbar, verständlich und unverwechselbar sein. ...

Themenvorschläge II

5. Patterns und Idiome, erprobte OOP-Designmuster und -Sprechweisen:
   Singleton, Container, Iterator, Beobachter,
   Copy-on-write, Barton-Nackman trick, Object-Generator, ...
6. Typsichere Sprachkonstrukte
   new statt malloc, const/constexpr statt #define,
   scoped enums, variants, sequence containeers, ...
7. Module

```
import std.io; // make names from std.io available
module M; // declare module M
export import std.random; // import and export names from std.random
export struct Point { // define and export Point
  int x;
  int y;
};
```

[1] https://de.wikipedia.org/wiki/Programmierstil (24. April 2017)

[2] https://en.wikipedia.org/wiki/The_Elements_of_Programming_Style#Lessons (24. April 2017)

[3] http://uwe-sauerland.de/richtlinien/Programmierstil.html (24. April 2017)

Index