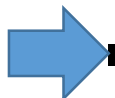


Inhalt der Vorlesung

- Einführung
- Kommunikation
- Konfiguration Management
- Software Qualität
 - Einführung
 - Software Fehler
 - ➡ ■ Konstruktive Qualitätssicherung
 - Software Tests
 - Statische Analyse
- Software Architektur und – Design
- Vorgehensmodelle
- Requirements Engineering



- 
- Software Richtlinien
 - Typisierung
 - Vertragsbasierte Programmierung
 - Fehlertolerante Programmierung
 - Portabilität
 - Dokumentation



Richtlinien regeln den Gebrauch einer Programmiersprache über die eigenen syntaktischen und semantischen Regeln hinaus.

Motivation:

- Vereinheitlichung
- Fehlerreduktion



Software Richtlinien

- 
- [Notationskonventionen](#)
 - [Sprachkonventionen](#)



Notationskonventionen werden auf verschiedenen Ebenen definiert (Projekt, Sprache, Betriebssystem,..)

Typischerweise betroffen:

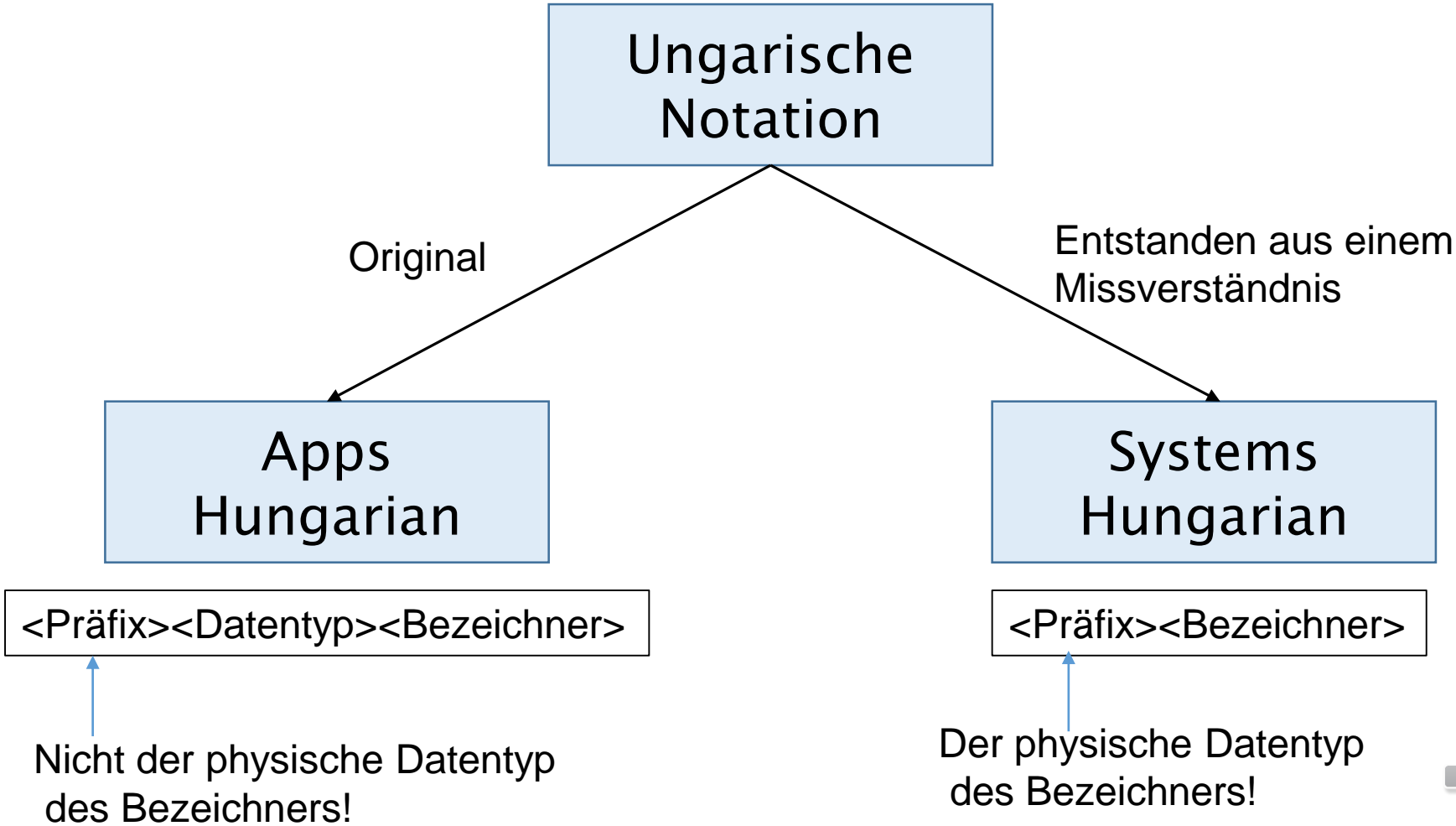
- Auswahl und Schreibweise von Bezeichnern
- Einrückungen, Verwendung von Leerzeichen
- Aufbau von Kontrollstrukturen
- Dokumentation



- **Pascal Case:** Bezeichner, sowie jedes enthaltene Wort startet mit Großbuchstaben.
- **Camel Case:** Bezeichner startet mit Kleinbuchstaben, jedes weitere Wort groß.
- **Uppercase:** Komplett in Großbuchstaben
- **Lowercase:** Komplett in Kleinbuchstaben



Ungarische Notation



Jeder Variablenname besteht aus zwei Teilen:

- **Präfix:** abkürzende Schreibweise für den Datentyp
- **Qualifier:** frei gewählter Name

Bsp:

Button butOK;
ListBox lbColorSelector;
CheckBox cbRemindMe;

Problem: Verstoß
gegen das Single
Source Prinzip

Linus Torvalds:

Encoding the type of a function into the name (so called Hungarian notation) is brain-damaged – the compiler knows the types anyway and can check those, and it only confuses the programmer. No wonder Microsoft makes buggy programs.



Präfix: Nimmt Bezug auf die Funktion der Variablen

Beispiele:

i – index in einem Array

p – Pointer

h – handle (pointerpointer)

c – Anzahl von Elementen (z.B. in einem Array)

rg – Ein durch integer indiziertes Array

gr – Verbund von Variablen (z.B. bei einer struct)

Siehe z.B. https://de.wikipedia.org/wiki/Ungarische_Notation



Datentyp: Einige Basetypes definiert

Beispiele:

f – Boolescher Datentyp (Bedeutung- nicht der physische Datentyp)

ch – ein Ein-Byte Zeichen

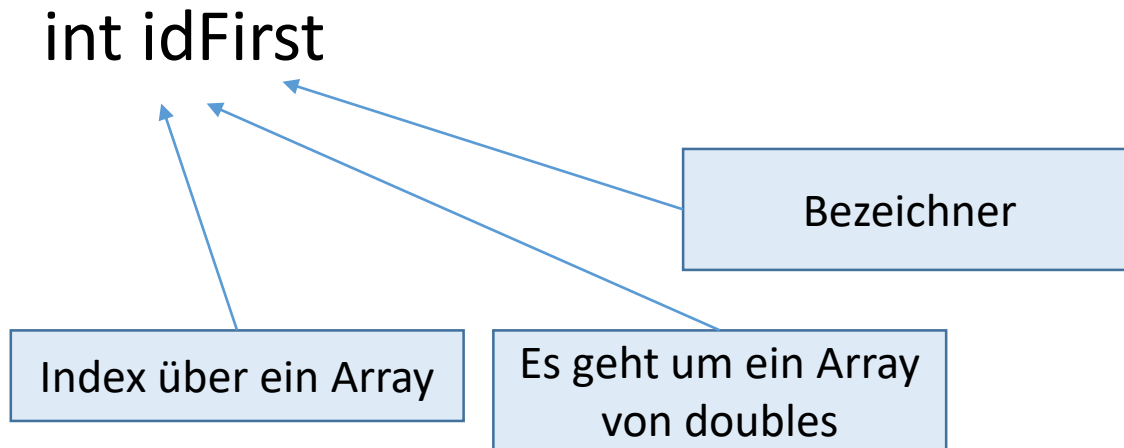
sz – ein Null terminierter String

Siehe z.B.

https://de.wikipedia.org/wiki/Ungarische_Notation



Beispiel



Coding Style

Die folgenden Coding Styles enthalten Notationskonventionen, gehen aber deutlich darüber hinaus.

C# Coding Style:

- siehe <https://msdn.microsoft.com/de-de/library/ff926074.aspx>

Java Coding Style: verschiedene Varianten: z.B.

- <https://google.github.io/styleguide/javaguide.html>
- <http://www.ambyssoft.com/essays/javaCodingStandards.html>
- Siehe auch: *Michael Inden, Der Weg zum Java Profi, Seite 1160*

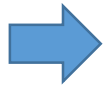
GNU Coding Standards for C:

- https://www.gnu.org/prep/standards/html_node/Writing-C.html#Writing-C



Software Richtlinien

- [Notationskonventionen](#)



- [Sprachkonventionen](#)



Bisher: Layout und Syntax,

Jetzt: Semantische Besonderheiten einer Sprache

Bsp: MISRA-C: Programmierstandard ➔ <http://www.misra-c.com>

(MISRA: Motor Industry Software Reliability Association)

Der MISRA-C-Programmierstandard definiert eine Untermenge des Sprachumfangs von C, d.h. er umfasst Richtlinien die zu einer Qualitätssteigerung (insbesondere der Softwarequalitätsaspekte der Zuverlässigkeit und Wartbarkeit) in der Software-Entwicklung führen sollen.



Hintergrund von MISRA

Weite Verbreitung von C gerade auch in sicherheitskritischen Bereichen,

ABER

- Verhalten teils undefiniert.
- C macht es leicht, die Sprache falsch zu gebrauchen.
- C erlaubt schwer verständliche Konstrukte.
- C überlässt dem Entwickler die Fehlerbehandlung zur Laufzeit.

➔ **MISRA unterstützt Entwickler bei der Entwicklung speziell sicherheitskritischer Systeme**

Vision von MISRA

The MISRA C Guidelines define a subset of the C language in which the opportunity to make mistakes is either removed or reduced.



1. Empfehlungen zu

- Tool Selection
- Projekt Aktivitäten
- Implementierung der MISRA Compliance

2. Guidelines (Directives and Rules)



Beispiele für MISRA Richtlinien

- Konstanten in einem vorzeichenlosen Kontext müssen mit einem U-Suffix versehen werden.
- Variablen vom Typ float (Gleitkommazahlen) sollen nicht mit den Vergleichsoperatoren == oder != getestet werden.
- goto soll nicht verwendet werden.
- magic numbers vermeiden und stattdessen sinnvoll benannte Konstanten verwenden: #define MAXSIZE 12.
- Division durch null verhindern: if (b!=0) a/=b;
- Compilerunabhängigkeit sicherstellen, z. B. shiften neg. Zahlen: $-3 \ll 4 \implies -3 * (1 \ll 4)$
- Operatorrangfolgen sind nicht trivial, daher Klammern verwenden: $(a \ \&\& \ b \ || \ c) \implies ((a \ \&\& \ b) \ || \ c)$.
- Rekursion darf in keiner Form auftreten.



Soziale Aspekte bei der Durchsetzung von Konventionen durch Codereviews

- Diejenigen, die die Konventionen am besten kennen, sind die Buhmänner.
- Der Gereviewte fühlt sich persönlich angegriffen.
- Der Reviewer will deswegen nichts mehr anmerken.

➔ Empfehlungen zum Vorgehen siehe nächste Seite

Durchsetzung von Konventionen

- Alle Beteiligten an der Erarbeitung der Konventionen mitarbeiten lassen.
- Konventionen gemeinsam weiterentwickeln.
- Begründete Ausnahmen akzeptieren (und dokumentieren).
- Toolgestützt prüfen (sowohl am Arbeitsplatz als auch beim Build Prozess).
- Regeln *zu Beginn* vereinbaren.
- Bestehende Regeln nur mit Bedacht ändern.