Wichtige Punkte für Abnahmen von Aufgaben im Praktikum Informatik 1 (C Programmierung)

Neben der reinen Programmfunktionalität sollten insbesondere folgende, teils auch nicht funktionale Aspekte bei der Abnahme der Praktikumsaufgaben geprüft werden.

Ab Beginn bzw. Kap. 5:

1.) Coding Rules

- Dateikopf, gut lesbare Formatierung, Leerzeilen zur Gliederung, Einrücken (nach **{**)
- Sinnvolle Variablennamen, Leerzeichen um binäre Operatoren
- Schreibweisen von Variablen (klein), #define Konstanten (GROSS) usw. beachten
- Keine numerischen Konstanten (magic Numbers; ausser 0 und 1), sondern symbolische Namens-Konstanten (#define) oder Kommentierung von Konstanten

Überprüfung der Compiler-Warnungen; Compilierung (weitgehendst) ohne Warnungen (mindestens mit –Wall, möglichst auch noch –O (bei Code-Blocks einstellen unter Settings – Compiler)

Ab while Schleifen, Kap. 16, (bereits ab if Anweisung, Kap. 11, erwünscht, mit Programmabbruch):

2.) Konsistenzprüfung von Eingaben; definiertes Programmverhalten auch bei Fehleingaben, wie z.B. Buchstaben statt Zahlen

Dazu: (siehe auch Kap. 16, Vorlesungs-Folien 62, 63 und 86)

- Ggf. Vorinitialisierung einzulesender Variablen (bei Fehleingaben wird Variable i.A. nicht gelesen bzw. beschrieben, sodass ohne Initialisierung mit undefiniertem Wert weitergearbeitet würde)
- Ggf. Abfrage des Rückgabewertes bei scanf() u.ä.; dummy getchar()

Programm-Test auch mit expliziten Fehleingaben (z.B. Buchstabe statt Zahl u.ä.); dabei darf kein undefiniertes Verhalten auftreten; Aufforderung zur wiederholten Eingabe oder Programmabbruch
(„Gorilla proved“; auch unsinnigste Eingaben müssen immer zu einem definierten Verhalten, z.B. zumindest zu einem definierten Programmabbruch führen)

Ab Funktionen, Kap. 22:

3.) Programm muss aus mindestens 2 Funktionen bestehen

- Eigentliche Aufgabe sollte in separater Funktion gelöst werden (i.A. auch mit Prototyp)
- main() Funktion sollte i.w. die zugehörige Testfunktion mit Ein- / Ausgaben u.ä. darstellen

Ab Arrays, Kap. 25:

4.) Bei Arrayzugriffen ist grundsätzlich sicherzustellen, dass keine Längenüberschreitung stattfindet; Konsistenz der Array-Länge muss immer, auch bei späteren Änderungen, gewährleistet sein

Daher: (siehe auch Kap. 25, Folie 55)

- Arraygrössen sind immer mit symbolischen Namens-Konstanten (z.B.: #define LEN_ARR … o.ä.) zu definieren
- Bei Zugriffen in Schleifen o.ä. über Index muss Überprüfung der Grösse / Länge immer mit symbolischen Namens-Konstanten erfolgen (Länge +/- 1 beachten)

Arrays sind statisch zu definieren, dynamische (C99) Arrays sind **nicht** erlaubt !

Programm-Test auch mit expliziten Fehleingaben (mehr als Array-Grösse u.ä.); dabei darf kein undefiniertes Verhalten auftreten, bzw. nicht über Array-Grenze geschrieben werden; Sicherheitslücke Pufferüberlauf !

15.03.2016, Herold, Lurz, Zwanger