Programmierrichtlinien zu den Übungen SWE I im WS2004

Richtlinien zur Programmierung, braucht man das wirklich?
Ja, denn wer schon einmal versucht hat, eigenen Quelltext zu verstehen, den er irgendwann einmal geschrieben hat, der wird dankbar über jeden Kommentar und jede verständliche Variable sein. Und bei fremdem Quelltext ist dies noch wichtiger. Einfach nur drauflos zu programmieren, funktioniert somit nicht.
Sinnvolle Variablenbenennung, die richtige Menge an Kommentaren, sowie saubere Formatierung des Quelltextes sollte man sich von Anfang an angewöhnen. Damit erspart man sich so manche lange Nacht vor dem Monitor.
Dieses Dokument wird laufend erweitert!

Allgemeines

Lesbarkeit und Verständlichkeit sind mindestens so wichtig wie die Programmgröße und die Ausführungsgeschwindigkeit.
Jede Zeile Quellcode enthält höchstens eine Anweisung.
Zeilen sollten eine Länge von 80 Zeichen nicht überschreiten. Zu lange Ausdrücke werden in mehrere Zeilen umgebrochen. Folgende Zeilen werden eine Ebene eingerückt.
Vor einem Komma wird kein Leerzeichen, nach einem Komma immer ein Leerzeichen eingefügt.
Vor und nach Operatoren wird ein Leerzeichen gesetzt. (Ausnahme: Bei Referenzierungs- und Dereferenzierungsoperatoren wird nach dem Operator kein Leerzeichen gesetzt.)

Benennungen

Dateinamen und Compiler

C++ Programme bekommen die Dateiextension `.cpp`. Wir werden in den Übungen SWE1 den Compiler `g++` verwenden. Alternativ kann auch der Compileraufruf mit `c++` erfolgen. Hilfe zum Compiler erhält man mit dem Kommando `man g++`.

Programmtexte

Programmtexte werden in englischer Sprache geschrieben. Die gilt insbesondere für die Bezeichnerwahl.

Kommentare

Kommentare werden nur in englischer Sprache geschrieben. Dies vor allem deswegen, weil Programme oft in internationaler Zusammenarbeit erstellt werden.

Schreibweisen

Klassen- und Typnamen beginnen mit einem Großbuchstaben, Funktionen und Methoden mit einem Kleinbuchstaben.
Konstanten werden in Großbuchstaben geschrieben.
Bei Bezeichnern, die aus mehreren Worten bestehen, wird jeder Wortanfang groß geschrieben: z. B. `setWindowColor`.
Bei Konstanten wird zwischen den Worten ein Unterstrich eingefügt:
z. B. `MAX_LEN`. Ansonsten sind Unterstriche in Bezeichnern zu vermeiden.
Funktionen und Methoden ohne Rückgabewerte beginnen mit einem Verb im Imperativ: z. B. `setColor`.
Funktionen und Methoden mit einem `bool`'schen Rückgabewert beginnen mit `is` : z. B. `isEmpty()`.

Visuelle Gestaltung

Kommentare

Es werden grundsätzlich nur einzeilige Kommentare verwendet (// statt /* ... */).

Dateibeginn

Am Anfang jeder Quelldatei soll ein Kommentar in der folgenden Form geschrieben werden:

```
//
// programme: Name des Programmes
// author : Namen des Autors
// date: Datum
// version : Versionsnummer
// purpose: kurze Beschreibung des Programmzweckes
//
```

Die Versionsnummer hat folgendes Format `x.yz`. Versionsnummern werden nur an Programme vergeben, die weitergegeben oder veröffentlicht werden. Es ist nicht vorteilhaft, bei jedem gefundenen Fehler die Nummer zu ändern.
Die Zahlen `x,y,z` werden immer um eins erhöht. Ändert sich `x`, werden `y` und `z` auf Null gesetzt. Ändert sich `y`, wird `z` auf Null gesetzt.
Die Zahl `x` kennzeichnet die Hauptversion und wird nur erhöht, wenn sich in der Benutzerführung Wesentliches ändert.
Die Zahl `y` zeigt kleinere Verbesserungen/Erweiterungen in der Benutzerführung an.
Die Zahl `z` zeigt interne Verbesserungen, Fehlerkorrekturen usw. an.
Optional kann hinter der Versionsnummer noch `alfa` oder `beta` stehen. Die Bezeichnung `alfa` zeigt, dass der Programmierer das Programm als im Wesentlichen fertig und fehlerfrei bezeichnet, `beta` bedeutet, dass dem Programmierer keine Fehler bekannt sind.

Einrückungen
Einrückungen sind mit Hilfe von Tabulatoren zu setzen. Der Tabulator wird auf zwei Leerschläge gesetzt.

Klammern
Bei komplexen Ausdrücken sind Klammern zu verwenden. Klammern um Ausdrücke werden innen und außen mit Leerzeichen gesetzt. Bei Funktionen steht die erste Klammer direkt nach dem Funktionsnamen.

Blöcke
Blöcke werden in einer der beiden international üblichen Versionen geschrieben:

```
if ( a == b )       // as in the lecture!
{
  return 0;
}

if ( a == b ) {     // as often in OO prg
  return 0;
}
```

Der Programmtext wird dann innerhalb des Blockes eingerückt.

Funktionen
Funktionen sind in folgender Form zu programmieren:

```
// purpose of function: ...
// parameter value: ...
// return value: ...
type1 functionname( type2 name2, type3 name3 )
{
  ...
}
```

Variablen
Variablen werden am Beginn eines Blockes definiert. Die einzige Ausnahme dazu stellen Zähl- oder temporäre Variablen dar, die nur in **engster** Nachbarschaft zur Definition verwendet werden.

Verbotenes
Kein `goto`!