Verlässliches Programmieren in C/C++ Coding Conventions

Projektgruppe 3, SoSe 2011

3. Juni 2011

1 Allgemeines

1.1 Sprache

Sprache für Bezeichner und Kommentare ist Englisch.

1.2 Zeichencodierung

Als Zeichencodierung kommt UTF-8 zum Einsatz.[1]

1.3 Zeilenumbrüche

Zeilenumbrüche werden im Unix-Standard gespeichert.[2]

1.4 Tabs

Zur Einrückung werden Tabs genutzt, die nicht durch Leerzeichen ersetzt werden. Die Tabgröße ist 4.

[1] Einstellungen -> General -> Workspace bzw. Dateieigenschaften -> Resource
[2] Einstellungen -> General -> Workspace

1.5 Dateien

Für jede Klasse wird ein Dateipaar aus [Klassenname].h und [Klassenname].cpp angelegt[3]. In diesen Dateien werden ausschließlich zur jeweiligen Klasse gehörende Codefragmente abgelegt.

2 Bezeichner

2.1 Variablen

Variablennamen bestehen aus Kleinbuchstaben, außer sie sind aus mehreren Wörter oder Wortteilen zusammengesetzt. Dann beginnt mit Ausnahme des ersten jedes Teilwort mit einem Großbuchstaben (CamelCase-Schreibweise). Unterstriche werden zur Abgrenzung eines Index verwendet.

Beispiele: myExampleVar, ip_server, ip_client

2.2 Konstanten

Konstanten werden in Großbuchstaben bezeichnet. Mehrere Wörter werden durch Unterstriche getrennt.

Beispiele: CONNECTION_SUCCESS, CONNECTION_FAILURE

2.3 Klassen

Klassennamen beginnen mit einem Großbuchstaben und bestehen ansonsten aus Kleinbuchstaben. Sind sie aus mehreren Wörtern oder Wortteilen zusammengesetzt wird wie bei den Variablen die CamelCase-Schreibweise genutzt.

Beispiele: Node, ExampleNode

2.4 Methoden und Funktionen

Für Methoden und Funktionen gelten die gleichen Regeln wie für Variablen (Kleinschreibung und CamelCase).

Beispiele: foo(), doSomething()

[3] Es empfiehlt sich die Nutzung des Menüs New -> Class in Eclipse

3 Standard-Sprachkonstrukte

Bei der Nutzung von Standard-Sprachkonstrukten sind Einrückungen zur Verbesserung der Übersichtlichkeit zu verwenden. Außerdem sind die kompakten Schreibweisen bzgl. Zeilenumbrüchen und geschweiften Klammern zu nutzen.

3.1 Verzweigungen und Schleifen

Beispiele:

```
if (var1 == var2)
    doSomething();

if (var1 == var2) {
    doSomething();
}

if (var1 > var2) {
    doSomething();
} else if (var1 < var2) {
    doSomthingElse();
} else {
    doSomeTotallyDifferentThing();
}

while (var1 > var2) {
    doSomething();
}

for (i=0; i<j; ++i) {
    doSomething(i);
}
```

3.2 Switch-Statements

Beispiele:

```
switch (i) {
case 1:
    doOne();
    break;
case 2:
```

```
    doTwo();
    break;
default:
    doDefault();
    break;
}
```

3.3 Klassen

Beispiel:

```
class example {
public:
    example();
    virtual ~example();
private:
    int internal(int i);
    double d;
};
```

3.4 Methoden und Funktionen

Beispiel:

```
int example::internal(int i) {
    // implement logic
}
```

4 Kommentare (insb. Doxygen)

Alle Klassen, Methoden/Funktionen sowie Attribute/Variablen sind gemäß der Doxygen-Vorgaben zu kommentieren. Im Folgenden sind einige Vorlagen gegeben.

4.1 Dateien / Klassen

Beispiel:

```
/**
 * Beschreibung der Klasse
 *
```

```
 * @author Max Mustermann <mustermann@example.com>
 * @since YYYY-MM-DD
 */
class foo {
    // ...
}
```

4.2 Methoden und Funktionen

Falls der Autor vom im Dateikopf genannten Autor abweicht darf auch im Kommentar zur Methode/Funktion ein @author-Tag genutzt werden.

Beispiel:

```
/**
 * Beschreibung der Funktion
 *
 * @param var1  Beschreibung von var1
 * @param var2  Beschreibung von var2
 * @return Beschreibung des Rueckgabewertes
 */
int foo(int var1, int var2) {
    // ...
}
```

4.3 Attribute

Der Kommentar für Attribute wird hinter dem Attribut selbst notiert, sodass der Kommentar um ein „<“ ergänzt werden muss.

Beispiel:

```
class foo {
public:
    int bar; /**< Beschreibung der Variable */
}
```

5 Sonstiges

5.1 Initialisierung von Zeigern

Bei der Initialisierung von Zeigern wird der Datentyp mit dem Stern versehen, nicht die Variable.

Beispiel:

```
int* foo; // nicht: int *foo;
```