Anhang A

Coding Style

A.1 Eine kurze Stilberatung für Programmierer

Während der Beschäftigung mit der Programmiersprache C werden Sie feststellen, dass es einige Regeln gibt die Sie unbedingt beachten müssen, während andere Regeln und Designentscheidungen eher als eine Art stille Übereinkunft zwischen Programmierern getroffen werden und als die 'übliche' Art und Weise der Programmierung angesehen werden.

Viele diese Regeln sind willkürlich getroffen, trotzdem ist es sinnvoll und vorteilhaft diese Konventionen zu kennen und sich daran zu halten, weil sie ihre Programme für Sie und andere einfacher lesbar machen und ihnen helfen Fehler zu vermeiden. Es können im wesentlichen drei Arten von Regeln unterschieden werden:

Naturgesetze: Diese Art von Regeln beschreiben Prinzipien der Logik und der Mathematik und gelten damit ebenfalls für Programmiersprachen wie C (oder andere formale Systems). So ist es zum Beispiel nicht möglich die Lage und Größe eines Rechtecks in einem Koordinatensystem durch weniger als vier Angaben genau zu beschreiben. Ein weiteres Beispiel besagt, dass die Addition von zwei natürlichen Zahlen dem Kommutativgesetz unterliegt. Dieser Zusammenhang ergibt sich aus der Definition der Addition und hat nichts mit der Programmiersprache C zu tun.

Regeln von C: Jede Programmiersprache definiert syntaktische und semantische Regeln die nicht verletzt werden dürfen, da sonst das Programm nicht korrekt übersetzt und ausgeführt werden kann. Einige dieser Regeln sind willkürlich gewählt, wie zum Beispiel das `=` Symbol, dass den Zuweisungsoperator darstellt und *nicht* die Gleichheit der Werte. Andere Regeln widerspiegeln die zugrundeliegenden Beschränkungen des Vorgangs der Kompilation und Ausführung des Programms. So müssen zum Beispiel die Typen der Parameter von Funktionen explizit spezifiziert werden.

Stil und Übereinkunft: Weiterhin existieren eine Reihe von Regeln die nicht durch den Compiler vorgegeben oder überprüft werden, die aber trotzdem wichtig dafür sind, dass Programme fehlerfrei erstellt werden, gut lesbar sind und durch Sie selbst und durch andere modifiziert, getestet und erweitert werden können. Beispiele dafür sind Einrückungen und die Anordnung von geschweiften Klammern, sowie Konventionen über die Benennung von Variablen, Funktionen und Typen.

In diesem Abschnitt werde ich kurz den Programmierstil zusammenfassen, der in diesem Buch verwendet wird. Er lehnt sich lose an die "Nasa C Style Guide"[1] an und das Hauptaugenmerk ist dabei auf die gute Lesbarkeit des Codes gerichtet. Es kommt weniger darauf an Platz zu sparen oder den Tippaufwand zu minimieren.

Da C eine - für eine Programmiersprache - vergleichsweise lange Geschichte aufweist, haben sich mehrere verschiedene Programmierstile herausgebildet. Es ist wichtig, dass Sie diese Stile lesen und verstehen können und dass Sie sich in ihrem eigenen Code auf einen Stil festlegen. Das macht den Programmcode viel zugänglicher, sollten es einmal notwendig werden, dass Sie den Code mit anderen Programmierern austauschen oder auf Teile ihres Codes zugreifen wollen, den Sie selbst vor einigen Jahren geschrieben haben.

A.2 Konventionen für Namen und Regeln für die Groß- und Kleinschreibung

Als generelle Regel sollten Sie sich angewöhnen bedeutungsvolle Namen für ihre Variablen und Funktionen zu verwenden. Idealerweise können Sie durch die Verwendung so genannter *sprechender Bezeichner* für Funktionen und Variablen bereits deren Verhalten und Verwendung erkennen.

Auch wenn es vielleicht aufwändiger ist eine Funktion `FindSubString()` anstatt `FStr()` zu nennen, so ist doch der erste Name fast selbsterklärend und kann ihnen eine Menge Zeit bei der Fehlersuche und späteren Wiederverwendung des Programms sparen.

Benutzen Sie keine Variablennamen die nur aus einem Buchstaben bestehen!

Ähnlich wie bei Funktionen sollten Sie die Namen ihrer Programmvariablen für sich selbst sprechen lassen. Durch einen geeigneten Namen wird von selbst klar welche Werte in der Variable gespeichert werden.

Wie zu jeder guten Regel gibt es auch hier einige Ausnahmen: Programmierer benutzen üblicherweise `i`, `j` und `k` als Zählvariablen in Schleifen und für räumliche Koordinaten werden `x`, `y` und `z` genutzt.

[1] www.scribd.com/doc/6878959/NASA-C-programming-guide

Benutzen Sie diese Konventionen wenn Sie in ihr Programm passen. Versuchen Sie nicht eigene, neue Konventionen zu erfinden, die nur Sie selbst verstehen.

Die folgenden Regeln zur Groß- und Kleinschreibung sollten Sie für die verschiedenen Elemente in ihrem Programm nutzen. Durch die einheitliche Verwendung eines Stils können Sie als Programmierer und Leser eines Programms sehr schnell die Bedeutung und Verwendung der verschiedenen Elemente bestimmen.

variablenNamen: Namen von Variablen werden immer klein geschrieben. Zusammengesetzte Namen werden dadurch gekennzeichnet, dass der erste Buchstabe des folgenden Worts groß geschrieben wird.

KONSTANTEN: verwenden ausschließlich Großbuchstaben. Um Konflikte mit bereits definierten Konstanten aus Bibliotheksfunktionen zu vermeiden kann es notwendig sein einen Prefix wie zum Beispiel `MY_CONSTANT` zu verwenden.

FunktionsNamen: beginnen immer mit einem Großbuchstaben und sollten nach Möglichkeit ein Verb enthalten welches die Funktion beschreibt (z.B. `SearchString()`). Funktionsnamen für Testfunktionen sollten mit '`Is`' oder '`Are`' beginnen (z.B. `IsNumber()`).

NutzerDefinierteTypen_t: enden immer mit einem '`_t`'. Namen für Typen müssen groß geschrieben werden. Dadurch werden Konflikte mit bereits definierten POSIX Namen vermieden.

pointerNamen_p: um Pointer Variablen sichtbar von anderen Variablen zu unterscheiden sollten Sie Pointer mit einem '`_p`' enden lassen.

A.3 Klammern und Einrückungen

Die größte Vielfalt der Stile finden sich in C bei der Positionierung von Klammern und Einrückungen. Deren Hauptaufgabe besteht darin den Code optisch zu gliedern und funktionale Bereiche durch die konsistente Verwendung von Einrückungen sichtbar voneinander abzugrenzen.

Die einzelnen Stile unterscheiden sich hierbei in der Art und Weise wie die Klammern mit dem Rest des Kontrollblocks positioniert und eingerückt werden. In diesem Kurs wird der so genannte *BSD/Allman* Stil verwendet, weil er den lesbarsten Code produziert. Bei diesem Stil nimmt der geschriebene Code mehr horizontalen Raum ein als bei dem ebenfalls sehr weit verbreiteten K&R Stil. Der *BSD/Allman* Stil macht es allerdings sehr viel einfacher alle öffnenden und schließenden Klammern im Blick zu behalten.

Im folgenden sehen Sie ein Auflistung verschiedener gebräuchlicher Klammer- und Einrückungsstile. Die Einrückungen betragen immer vier Leerzeichen pro Level:


```
/*Whitesmiths Style*/
   if (condition)
       {
       statement1;
       statement2;
       }
```

Der Stil ist nach einem frühen kommerziellen C Compiler *Whitesmiths C* benannt, welcher diesen Stil in seinen Programmbeispielen verwendet hat. Die Klammern befinden sich auf dem äußerem Einrückungsniveau.

```
/*GNU Style*/
   if (condition)
     {
       statement1;
       statement2;
     }
```

Die Klammern befinden sich in der Mitte zwischen inneren und äußerem Einrückungsniveau.

```
/*K&R/Kernel Style*/
   if (condition) {
       statement1;
       statement2;
   }
```

Dieser Stil wurde nach den Programmierbeispielen des Buchs *The C Programming Language* von Brian W. Kernighan und Dennis Ritchie (die C-Entwickler) benannt.

Der K&R Stil ist am schwersten zu lesen. Die öffnende Klammer befindet sich an der äußersten rechten Seite der Kontrollanweisung und ist damit schwer zu finden. Die Klammern haben unterschiedliche Einrückungstiefen. Trotzdem ist dieser Stil weit verbreitet und viele C Programme nutzen ihn. Sie sollten deshalb in der Lage sein diesen Code lesen zu können.

```
/*BSD/Allman Style*/
   if (condition)
   {
       statement1;
       statement2;
   }
```

Die Klammern befinden sich auf dem inneren Einrückungsniveau und sind damit leicht zu finden und zuzuordnen. Dieser Stil wird für alle Beispiele dieses Kurses verwendet.

Wenn Sie Programme schreiben ist es am Wichtigsten sich auf einen Stil festzulegen und diesen Stil dann konsequent beizubehalten. In größeren Softwareprojekten sollten sich alle Mitwirkenden auf einen gemeinsamen Stil einigen.

Moderne Programmierumgebungen wie zum Beispiel Eclipse [2] machen es leicht durch automatische Einrückungen einen Stil durchzusetzen.

A.4 Layout

Kommentarblöcke können dazu genutzt werden die Funktion des Programms zu dokumentieren und zusätzliche Angaben zum Ersteller zu machen. Sinnvollerweise werden diese Angaben als erste Angaben noch vor den Funktionsdeklarationen vorgenommen.

Einen ähnlichen Kommentarblock können Sie vor jeder Funktion verwenden um deren Funktion zu beschreiben.

```
/*
 * File:     test.c
 * Author:   Peter Programmer
 * Date:     May, 29th, 2009
 *
 * Purpose: to demonstrate good programming
 *          practise
 * /

#include <stdio.h>
#include <stdlib.h>

/*
 * Main function, input: none, output: 'HelloWorld'
  */

int main (void)
{
    printf("Hello World!\n");
    return EXIT_SUCCESS;
}
```

[2] www.eclipse.org