Regeln für das Programmieren

Labor “Rechnerorganisation”
Berufsakademie Stuttgart
Fachrichtung Nachrichtentechnik

Ingo Phleps

Stand 2. Februar 2004

http://home.ntz.de/phleps/programming/prgregeln.pdf

Leitsätze:

> Ziel des Programmierens ist nicht nur ablauffähiger Code,
> sondern eine Lösung,
> die bei ähnlichen Aufgaben wieder verwendet werden kann.
> (Quelle nicht bekannt)

> Software, die nicht gewartet wird, verliert sehr rasch ihren Wert.
> **Software, die nicht wartbar ist, hat keinen Wert!**
> U. Kern

Die Beispiele dieser Regeln beziehen sich auf die Programmiersprache C.

Erst denken, dann eingeben:

- Machen Sie sich die Aufgabenstellung klar.
- Überlegen Sie sich zuerst einen passenden Lösungsweg, bevor Sie mit dem Kodieren beginnen!

Strukturiert programmieren:

- Spalten Sie die Aufgabe in Funktionsblöcke (Prozeduren) auf.
- Gliedern Sie die Lösung übersichtlich.
- Verwenden Sie niemals die `goto`-Anweisung!
- Halten Sie in Quelldateien folgende Reihenfolge ein:

```
/************************************************************
 * Programmkopf (Modul-, Dateikopf)
 ************************************************************/

# include <...>          /* allgemeine Headerdateien         */
# include "..."          /* projektspezifische Headerdateien */

# define ...             /* symbolische Konstanten           */

struct lt_...            /* Definitionen lokaler Datentypen  */

static ...               /* globale Variablen                */

...                      /* Prototypen für static-Funktionen */

...                      /* Funktionsdefinitionen mit        */
                         /* Funktionskopf und Code           */
```

Informationen konzentrieren:

- Realisieren Sie mehrfach auftretende Abläufe als Funktion oder Makro.
- Verwenden Sie symbolische Konstanten statt Zahlen.
- Stellen Sie Deklarationen (aber niemals Definitionen!!), die von mehreren Modulen verwendet werden, in Header-Dateien.

Informationen kapseln:

- Beschränken Sie, soweit möglich, den Gültigkeits- bzw. Sichtbarkeitsbereich
  - von Variablen auf die betreffende Funktion,
  - von Datentypen und Funktionen auf das Modul.
- Verwenden Sie globale Variablen nur dort, wo sie unbedingt notwendig sind.

Stellen Sie die Übereinstimmung zwischen Peripherie und Programmzustand sicher:

- Gehen Sie nicht davon aus, in welchem Zustand die Peripherie (z. B. Meßgerät, Schalter, ...) sein müßte, sondern stellen Sie den aktuellen Zustand fest (Abfrage oder definiert einstellen).

Gestalten Sie ihre Programme lesbar und verständlich:

- Verwenden Sie aussagekräftige Namen.
- Verwenden Sie in Namen keine Sonderzeichen (z. B. Umlaute).
- Halten Sie Konventionen ein (z. B. Groß- u. Kleinschreibung, ...).
- Deklarieren Sie für jede C-Funktion einen Funktionsprototypen nach ANSI-C. Er muß
  - den Funktionstyp (immer, auch bei Funktionen vom Typ `int`!) sowie
  - die Parametertypen und -namen enthalten und muß
  - vor der Funktionsdefinition stehen.
- Versehen Sie jedes Modul und jede Funktion mit einem Kopf, der die Aufgabe und die Schnittstelle des Moduls bzw. der Funktion beschreibt. Die für die Anwendung benötigten Informationen sollten daraus ersichtlich sein.
  Der Funktionskopf muß mindestens folgende Informationen enthalten:
  - Beschreibung jedes Funktionsparameters einschließlich der Angabe, ob die Funktion lesend und / oder schreibend darauf zugreift
  - Funktionsergebnis
  - Beschreibung, was die Funktion tut
  - Einschränkungen und Fallen beim Einsatz der Funktion
- Kommentieren Sie mindestens
  - jede symbolische Konstante
  - jede Variable
  - jeden Strukturblock
  - jede Schleife
  - jede Abfrage
- Wiederholen Sie im Kommentar nicht, was aus der Anweisung ersichtlich ist, sondern beschreiben Sie den übergeordneten Zusammenhang.
  Beispiel:

  Bitte so:
  ```
  /* Kein Fehler ? */
  if ( 0 == result )
  ```

  Nicht so:
  ```
  /* Ergebnis = 0 ? */
  if ( 0 == result )
  ```
- Beschreiben Sie nicht nur, *was* die Anweisungen tun, sondern bei Bedarf auch, *warum* sie notwendig sind.

Programmieren Sie defensiv:

- Verwenden Sie einen möglichst empfindlichen Warning-Level des Compilers.
- Beachten Sie auch Warnungen und beseitigen Sie deren Ursachen.
  Ausnahme: Die Warnung "*warning: 'rcsHeader' defined but not used*" für die Variable, mit der die RCS-Versionsnummer in den Objekt-Code übernommen wird, kann akzeptiert werden.
- Vermeiden Sie Programmiertricks!
  Tricks und unkonventionelle Methoden sollten nur dort verwendet werden, wo sie wesentliche Vorteile bringen. Voraussetzung ist eine ausführliche Beschreibung, wie das Verfahren funktioniert und warum es anstatt einer konventionellen Methode verwendet wurde.