WBS TRAINING

Programmierung(1)

UNSER ♥ SCHLÄGT FÜR BILDUNG

Agenda

- **Kategorisierung von Schleifen**
  - Ort der Bedingungsüberprüfung
    - Kopf- versus Fuß-gesteuert
  - Art der Bedingungsüberprüfung
    - Zähler- versus nicht-Zähler-gesteuert
- **Kriterien für die Auswahl von Schleifen**
  - Mindestens 1 Durchlauf versus (eventuell) kein Durchlauf
  - Potentielle Austauschbarkeit von Schleifentypen
- **Modulo-Operator**
  - Definition
  - Beispielaufgabe im PAP, Struktogramm, Pseudocode und ANSI C
- Ausführliches Training + Ergebnisbesprechung
- Fachpraktische Anwendungen

Kategorisierung von Schleifen – Kopf versus Fuß

- Wie bereits angedeutet, können wir Schleifen bzgl. des **Ortes** der Bedingungsüberprüfung unterscheiden. Falls diese Bedingungsüberprüfung VOR jedem Durchlauf (also im „Kopf“ der Schleife) geschieht, so spricht man von **kopf-gesteuerten** Schleifen.

  => WHILE- und FOR-Schleifen sind demnach kopf-gesteuert

- Entsprechend werden Schleifen, bei denen die Bedingungsüberprüfung NACH jedem Durchlauf geschieht als **fuß-gesteuert** bezeichnet.

  => DO-WHILE-Schleifen sind also fuß-gesteuerte Schleifen

- Für kopf-gesteuerte Schleifen ist charakteristisch, dass diese möglicherweise **keinen einzigen** Durchlauf vollziehen, da die Schleife bereits mit ihrer ersten Bedingungsüberprüfung (und also vor ihrem ersten Durchlauf) abgebrochen werden kann.
- Fuß-gesteuerte Schleifen werden hingegen stets **mindestens einen** Durchlauf absolvieren, da die Bedingungsüberprüfung immer erst nach einem Durchlauf stattfindet. (Eine fuß-gesteuerte Schleife kann also frühestens nach ihrem ersten Durchlauf abgebrochen werden.)

Kategorisierung von Schleifen – Zähler-gesteuert?

- Eine vom Ort der Bedingung unabhängige Kategorisierung von Schleifen gelingt, wenn man sich die Frage stellt, ob die Bedingung einer Schleife von einem Zählerwert spricht. Schleifen, für die dies gilt, werden **Zähler-gesteuert** genannt. (Der Wert des Zählers steuert dann also, ob die Schleife abgebrochen wird, oder ob ein weiterer Durchlauf starten soll)
- Zähler-gesteuerte Schleifen können prinzipiell kopf- oder fuß-gesteuert sein. Da jedoch ihr Zähler die **Anzahl der Durchläufe** bestimmt, und weil diese Anzahl im Einzelfall auch vom Wert 0 seien könnte, bieten sich in der Regel kopf-gesteuerte Schleifen an, die entsprechend auch schon vor dem ersten Durchlauf abgebrochen werden könnten.
- Bei Zähler-gesteuerten Schleifen ist die klassische Wahl des Schleifentyps sicherlich eine **FOR-**Schleife. Es wäre aber umgekehrt falsch, anzunehmen, dass jede FOR-Schleife Zähler-gesteuert sei. Tatsächlich werden wir im späteren Verlauf Beispiele kennenlernen, bei denen eine verwendete FOR-Schleife zwar mit einem Schleifenzähler arbeitet (also einen Startwert setzt und den Schleifenzähler nach jedem Durchlauf weiter-zählt) dabei allerdings eine Bedingung verwendet, die nicht vom Zähler abhängt. Durchlauf und Ende einer solchen Schleife werden demnach nicht vom Zähler gesteuert.

Kriterien für die Auswahl von Schleifen – Anzahl der Durchläufe

- Immer dann, wenn eine Schleife **mindesten 1-mal durchlaufen** werden soll, bietet sich eine fuß-gesteuerte Schleife an. Typische Beispiele sind:
  - Eingabe-Schleifen, bei denen die Korrektheit der Eingabe natürlich erst NACH dem ersten Durchlauf kontrolliert werden kann.
  - Fehleranfällige Funktionsaufrufe, bei denen erst nach dem Versuch kontrolliert werden kann, ob diese wie gewünscht gearbeitet haben.

- In **allen anderen Fällen** sollten kopf-gesteuerte Schleifen verwendet werden.
  - Falls diese ohne Zähler-Variable arbeiten, so sollten WHILE-Schleife gewählt werden.
  - Falls mit (einer oder mehreren) Zähler-Variablen gearbeitet wird, können FOR-Schleifen empfohlen werden. Dies gilt unabhängig davon, ob die jeweiligen Schleifen auch Zähler-gesteuert sind, da FOR-Schleifen eine kompakte Notation der Startwert-Initialisierung und des Weiterzähl-Modus ermöglichen.

Kriterien für die Auswahl von Schleifen – Potentielle Austauschbarkeit

- Unabhängig von den zuvor angesprochenen Empfehlungen gilt jedoch, dass jede der drei uns bekannten Schleifen-Typen durch jeden anderen ersetzt werden könnte.
- Dies werden wir aus Trainingsgründen bei einer kommenden Beispielsaufgabe praktisch vorführen. Dabei wird dann allerdings auch deutlich werden, dass die Ersetzung eines günstigen Schleifentyps (durch einen beliebigen anderen) gelegentlich recht aufwendig sein kann, und jedenfalls nicht selten sehr unnatürlich wirkt. Zudem leidet die Lesbarkeit des Quellcodes.

=> Eventuell kann durch diese Form von „Ausschluss-Kriterien“ noch deutlicher werden, warum man sich im Einzelfall für bestimmte (und gegen andere) Schleifen-Typen entscheidet.

Modulo-Operator – Definition

- Der Modulo-Operator ermittelt den Rest beim ganzzahligen Teilen … **Beispiele**:
  - 5 modulo 3 = **2**, denn 5: 3 = 1 Rest **2**
  - 25 modulo 7 = **4**, denn 25: 7 = 3 Rest **4**
  - 33 modulo 11= **0**, denn 33:11 = 3 Rest **0**

- Der Modulo-Operator bzgl. **n** ergibt einen Wert zwischen **0** und **n-1** … **Beispiel (für n=3)**:
  - 0 modulo 3 = 0
  - 1 modulo 3 = 1
  - 2 modulo 3 = 2
  - 3 modulo 3 = 0
  - 4 modulo 3 = 1
  - 5 modulo 3 = 2
  - 6 modulo 3 = 0
  - 7 modulo 3 = 1
  - …

Achtung:
Für beliebige ganze Zahlen x ist **x modulo 0** nicht zulässig!
(dies entspräche dem ebenfalls unzulässigen Teilen durch 0)

Modulo-Operator – Motivation

- Wer mit ganzen Zahlen rechnet, der vermeidet Rundungsfehler, die bei Berechnungen mit Kommazahlen in der Regel auftreten.
- Addieren, Subtrahieren und Multiplizieren von ganzen Zahlen ist im wesentlichen unproblematisch. Wer aber zwei ganze Zahlen **dividiert** und Kommawert-Ergebnisse vermeiden möchte, der muss eine **Division mit Rest** durchführen. Die beiden sich hierbei ergebenden Werte müssen auf zwei unterschiedliche Operationen verteilt werden:
  - Die **Division** ermittelt nur den **ganzzahligen Wert** und ignoriert den Rest: 13 : 4 = **3**
  - Der **Modulo** ermittelt nur den **Rest** und ignoriert den ganzzahligen Wert: 13 modulo 4 = **1**

- Tatsächlich gibt es aber auch weitere Motivationen, sich mit dem Modulo zu beschäftigen. Wir wollen zunächst nur die beiden folgenden vorstellen:
  - Ob eine ganze Zahl **n** durch einen Wert **t** teilbar ist, kann mit dem Modulo-Operator entschieden werden, denn in diesem Fall gilt: **n modulo t == 0**
  - Des weiteren werden wir den Modulo-Operator auch beim arbeiten mit Zufallszahlen nutzen können. Mit diesem auf den ersten Blick verblüffenden Zusammenhang beschäftigen wir uns Morgen … Vorfreude ;-)

Modulo-Operator – Beispielaufgabe

- Vom User wird zunächst eine beliebige ganze Zahl **a** abgefragt. Anschließend eine weiter ganze Zahl **b**, die ungleich 0 sein muss. (Beide Eingaben werden der Einfachheit halber nicht überprüft). Anschließend soll die folgende Ausgabe erscheinen, bevor das Programm endet:

  a „geteilt durch “ b „ergibt:“ a:b „Rest“ a modulo b

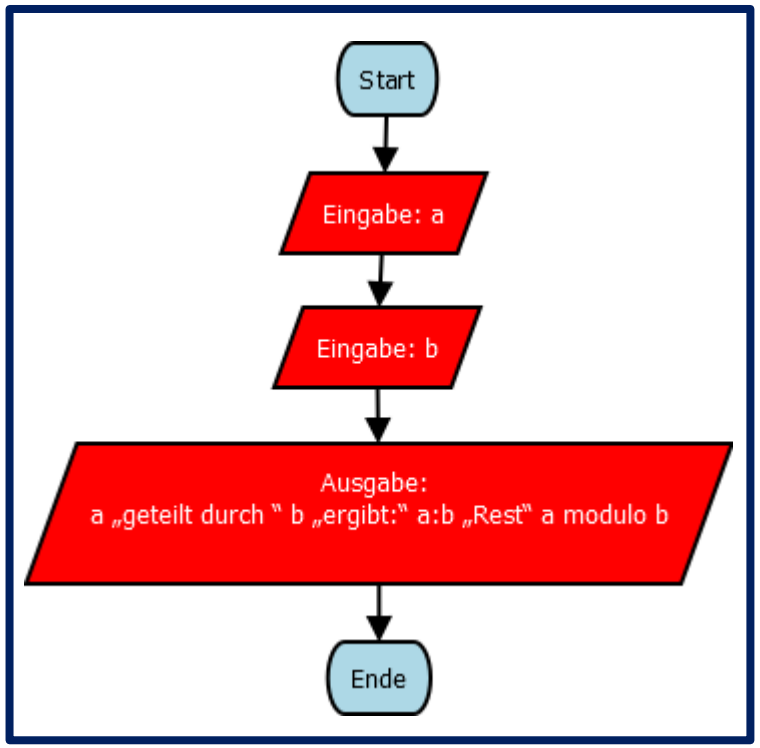
Hinweis:

Die Modulo-Operation besitzt kein eigenes Symbol in **PAP**´s, **Struktogrammen** oder **Pseudocodes**.

Die jeweiligen Notationsformen werden wir entsprechend auf der folgenden Folie gleichzeitig und ohne Mühe abhandeln können …

… interessant wird daher vor Allem die Syntax in **ANSI C** sein.

Beispielaufgabe – PAP, Struktogramm, Pseudocode



MODULO-Beispiel

Eingabe: a
Eingabe: b
Ausgabe: a "geteilt durch" b "ergibt" a:b "Rest" a modulo b

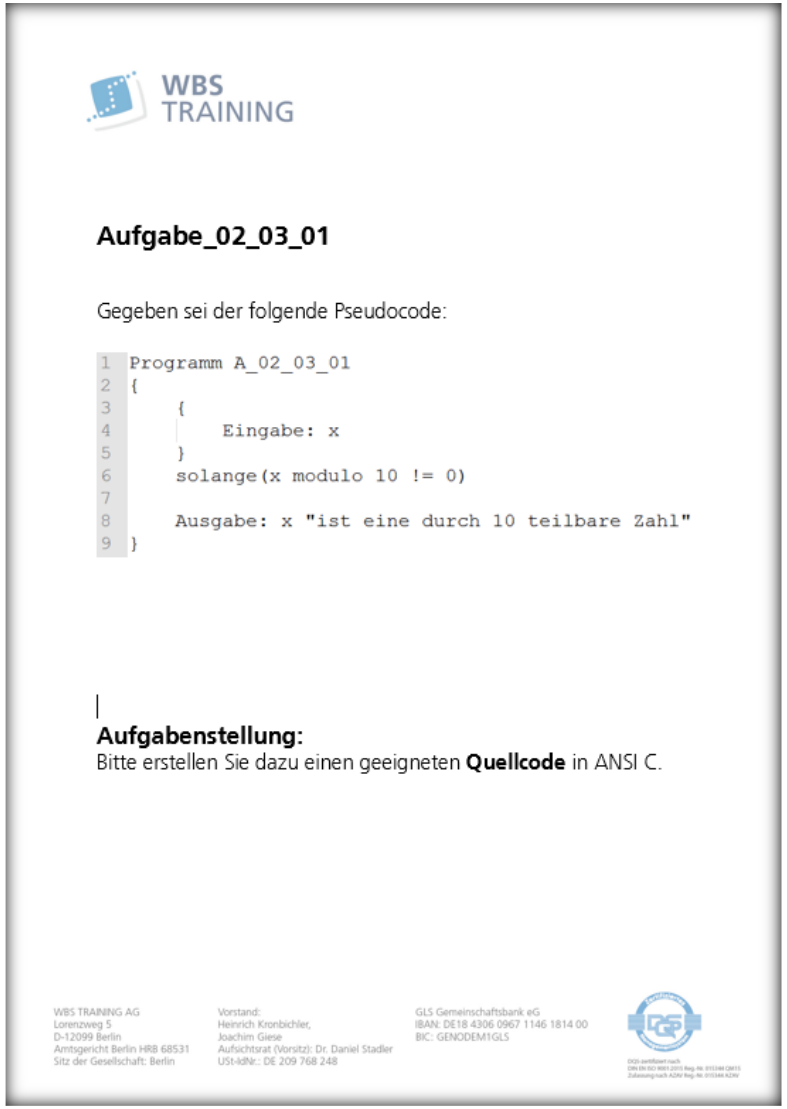
```
Programm Modulo-Beispiel
{
        Eingabe: a
        Eingabe: b
        Ausgabe: a „geteilt durch“ b „ergibt“ a:b „Rest“ a modulo b
}
```

WBS TRAINING

Beispielaufgabe – Quellcode – Division und Modulo

```
# include<stdio.h>

main()
{
        int a,b;

        printf("Geben Sie bitte eine beliebige ganze Zahl ein: ");
        fflush(stdin);
        scanf("%d",&a);

        printf("Geben Sie bitte eine ganze Zahl ungleich 0 ein: ");
        fflush(stdin);
        scanf("%d",&b);

        printf("%d geteilt durch %d ergibt: %d Rest %d",a,b,a/b,a%b);
}
```

Modulo-Operator – Gemeinsame Übung A_02_03_01



WBS TRAINING

Aufgabe_02_03_01

Gegeben sei der folgende Pseudocode:

```
1  Programm A_02_03_01
2  {
3      {
4          Eingabe: x
5      }
6      solange(x modulo 10 != 0)
7
8      Ausgabe: x "ist eine durch 10 teilbare Zahl"
9  }
```

Aufgabenstellung:
Bitte erstellen Sie dazu einen geeigneten **Quellcode** in ANSI C.

WBS TRAINING AG
Lorenzweg 5
D-12099 Berlin
Amtsgericht Berlin HRB 68531
Sitz der Gesellschaft: Berlin

Vorstand:
Heinrich Kronbichler,
Joachim Giese
Aufsichtsrat (Vorsitz): Dr. Daniel Stadler
USt-IdNr.: DE 209 768 248

GLS Gemeinschaftsbank eG
IBAN: DE18 4306 0967 1146 1814 00
BIC: GENODEM1GLS

WBS TRAINING

VIELEN DANK FÜR IHRE AUFMERKSAMKEIT!