

# Pseudocode, PAP, Struktogramm

## Vorgehensweise

- Aufgabenstellung lesen & verstehen
  - Was ist der Zweck des Programms?
  - Welche Eingaben, Verarbeitungen und Ausgaben gibt es?
- Eingaben Identifizieren
  - Was bekommt die Funktion als Parameter oder von außen übergeben?
- Verarbeitung logisch zerlegen
  - Rechenschritte, Bedingungen, Schleifen? Was passiert wann?
  - Gibt es Rechenvorschriften wie "abwechselnd mit 1 und 3 gewichten"? Notieren!
- Ergebnis/Ausgabe festlegen
  - Was soll am Ende zurückgegeben oder angezeigt werden?

## Aufgaben

Für den Onlineversand sollen die Weinflaschen mit Barcode-Aufkleber versehen werden, der nähere Informationen zum Wein enthält. Der Barcode besteht aus zehn Ziffern.

123-456-78-9-p

- Ziffern 1 bis 3: Schlüssel für Region
- Ziffern 4 bis 6: Schlüssel für Rebsorte
- Ziffern 7 - 8: Jahrgang
- Ziffer 9: Geschmacksangabe (lieblich, halbtrocken, trocken ...)
- Dazu kommt an der 10. Stelle eine Prüfziffer p.

Die Prüfziffer soll nach folgender Beschreibung errechnet werden:

- Die einzelnen Ziffern werden alternierend gewichtet von Links nach Rechts mit 1 und 3:
  - $\text{Ziffer}_1 \times 1$ ,  $\text{Ziffer}_2 \times 3$  ....  $\text{Ziffer}_9 \times 1$
- Die 9 gewichteten Produkte werden addiert.
- Die Prüfziffer ist die Differenz der Summe zum nächstkleineren Vielfachen von 10 (modulo 10)

```
function ermittlePrüfziffer(int[] barcode)
{
```

```

int pruefziffer = 0

für i = 0; i < barcode.laenge; i++
{
    wenn (i % 2 = 0)
    {
        pruefziffer += barcode[i] * 3
    } sonst
    {
        pruefziffer += barcode[i] * 1
    }
}

pruefziffer = (10 - (pruefziffer % 10)) % 10

rueckgabe pruefziffer
}

```

Alle aktuell vorhandenen Barcodes der Weine und deren Jahresabsatz sind in einer zweidimensionalen Tabelle "Absatz" in folgender Form gespeichert:

| Region | Rebsorte | Jahrgang | Geschmacksrichtung | Absatz in Stk. |
|--------|----------|----------|--------------------|----------------|
| 123    | 456      | 78       | 9                  | 46             |
| 333    | 125      | 20       | 4                  | 998            |
| ...    | ...      | ...      |                    |                |

Erstellen Sie auf der Folgeseite eine Funktion "sucheTopseller", die für ein übergebendes Kriterium (0 = Region; 1 = Rebsorte, 2 = Jahrgang; 3 = Geschmacksrichtung) und einen entsprechenden Vorgabewert (z.B. 123 für eine bestimmte Region) den umsatzstärksten Wein ermittelt und den Barcode dieses Weines als Zeichenkette ohne Prüfziffer zurückliefert. Das zweidimensionale Array "Absatz" steht in der Funktion "sucheTopseller" zur Verfügung. Gehen Sie davon aus, dass alle Weine einen unterschiedlichen, positiven Absatz haben.

Hinweis: Für das Zusammenfügen von Zeichenketten kann der + Operator verwendet werden. Gemischte Ausdrücke vom Typ String und Ganzzahl sind möglich.

```

FUNKTION sucheTopseller(int kriterium, int vorgabe) : STRING
    int barcode = 0;
    int maxAbsatz = -1;
    int maxIndex = -1

```

```

VON i = 0 BIS Absatz.laenge
  WENN Absatz[i][kriterium] == vorgabe und Absatz[i][4] > maxAbsatz //
    maxAbsatz = Absatz[i][4]
    maxIndex = i
  ENDE WENN
ENDE VON

barcode = barcode + Ansatz[maxIndex][i]
VON i = 1 BIS 3
  barcode = barcode + "-" + Absatz[maxIndex][i]
ENDE VON
RUECKGABE barcode
ENDE FUNKTION

```

Für die Helferlein e.V. soll eine Prozedur entwickelt werden, welche die erbrachten Leistungen und die Aufsummierten Entgelte auflistet. Die Daten sind in einem Journal gespeichert.

Journal (Beispiel)

| Datum      | MitgliedID | LeistungID | AnzahlStunfrn |
|------------|------------|------------|---------------|
| 01.04.2021 | 100062     | 100076     | 2             |
| 11.04.2021 | 100062     | 100076     | 3             |
| 10.04.2021 | 100062     | 500123     | 1             |
| 13.04.2021 | 201235     | 200234     | 1             |
| 14.04.2021 | 201235     | 200234     | 1             |
| 07.04.2021 | 201235     | 200356     | 1             |

Das Journal ist nach MitgliedID und bei gleicher MitgliedID nach LeistungID sortiert.

Die Ausgabeliste soll wie folgt aufgebaut sein:

| Nr | MitgliedID | Name    | Vorname | LeistungID | Leistung | AnzahlStunden | Stundensatz | Gesamt |
|----|------------|---------|---------|------------|----------|---------------|-------------|--------|
| 1  | 100062     | Clausen | Jens    | 100076     | Gießen   | 5             | 6,00        | 30,00  |
| 2  | 100062     | Clausen | Jens    | 500123     | Umgraben | 1             | 6,00        | 6,00   |
|    |            |         |         |            |          |               | Summe       | 36,00  |
| 1  | 201235     | Rader   | Sabrina | 200234     | Hausputz | 2             | 6,00        | 12,00  |
| 2  | 201235     | Rader   | Sabrina | 200356     | Einkauf  | 1             | 6,00        | 6,00   |

| Nr | MitgliedID | Name | Vorname | LeistungID | Leistung | AnzahlStunden | Stundensatz | Gesamt |
|----|------------|------|---------|------------|----------|---------------|-------------|--------|
|    |            |      |         |            |          |               | Summe       | 18,00  |
|    |            |      |         |            |          |               | Gesamtsumme | 54,00  |

Folgende Funktionen sollen verwendet werden:

|  |  |
|--|--|
| hole_satz() : String   | Liest den nächsten Datensatz der Journal-Tabelle in eine Zeichenkette ein. Kann kein Satz mehr gelesen werden, liefert die Funktion den String "". |
| lese_m_id(satz:string):Integer   | Ermittelt die MitgliedID aus Satz  |
| lese_l_id(satz:string):Integer   | Ermittelt die LeistungsID aus Satz   |
| lese_anz_std(satz: string):Integer   | Ermittelt die Anzahl der Stunden aus Satz  |
| schreibe_kopf()  | Schreibt die Kopfzeile der Positionen-Tabelle  |
| schreibe_datan (         nr: Integer,         mitgliedid: Integer,         leistungid: Integer,         anzahlstunden: Integer,         stundensatz: Double,         summe: Double       ) | Schreibt eine Datenzeile in der geforderten Darstellung. Name, Vorname und Leistung werden automatisch aus mitgliedid und leistungid ermittelt.    |
| schreibe_summe(summe: Double)  | Schreibt die (berechnete) Summe für ein Mitglied   |
| schreibe_gsumme(gsumme: Double)  | Schreibt die (berechnete) Gesamtsumme des Journals   |

Entwickeln Sie einen Algorithmus für die Prozedur `erstelle_liste(stundensatz: Double)`, der die Liste entsprechend der Vorgabe schreibt und dazu die entsprechenden Werte berechnet. Verwendete Variablen müssen nicht deklariert werden.

```
erstelle_liste(stundensatz: Double)
```

```
  printKopf()
```

```
  datensatz = hole_satz()
```

```
  zaehler = 0
```

```
  gesamtsumme = 0;
```

```
  SOLANGE datensatz != "" Dann
```

```
    mitglied = lese_m_id(datensatz)
```

```
    summeMitglied = 0
```

```
SOLANGE nächsterDatensatz != "" UND lese_m_id(datensatz) = mitglied
```

```
    leistung = lese_l_id(datensatz)
```

```
    stunden = 0;
```

```
    WENN lese_l_id(datensatz) = lese_l_id(nächsterDatensatz) DANN
```

```
        summeLeistung += lese_anz_stunden(datensatz) + lese_anz_stunden(nächsterDatensatz) *
```

```
    stundensatz
```

```
    schreibe_daten(zaehler, lese_m_id(datensatz), lese_l_id(datensatz), lese_anz_stunden(datensatz),  
    stundensatz, summeLeistung)
```

```
    schreibe_summe(summe)
```

```
    SONST
```

```
        summeLeistung += lese_anz_stunden(datensatz) + lese_anz_stunden(nächsterDatensatz) *
```

```
    stundensatz
```

```
    gesamtsumme = gesamtsumme + summeleistung
```

```
    ENDE WENN
```

```
    datensatz = naechsterSatz
```

```
    WENN get_m_id(datensatz) != get_m_id(naechsterDatensatz)
```

-----

Eine geplante Stored Procedure soll Folgendes leisten:

1. Zunächst werden alle Ferienwohnungen im selben Ort ermittelt und in einer Tabelle gespeichert
2. Jede Wohnung aus dieser Tabelle wird anschließend darauf überprüft, ob die Bettenzahl mindestens so groß wie bei der stornierten Wohnung ist und der Preis höchstens um 10% abweicht
3. Eine Wohnung, die diese Kriterien nicht erfüllt, wird aus der Tabelle gestrichen
4. Sollten anschließend mehr als 3 Wohnungen übrigbleiben, werden alle bis auf die drei preiswertesten gelöscht. (Hier ist der betreffende Preis zu ermitteln.)

5. Sollte keine Wohnung übrig bleiben, werden alle Wohnungen aus demselben Land wie die stornierte Wohnung ermittelt und für diese Schritt 2-4 wiederholt.
6. Sollte es keine Wohnung in demselben Land mehr geben, erfolgt eine Misserfolgsmeldung

Zeichnen Sie für diese Stored Procedure ein Struktogramm

---

Revision #3

Created 16 April 2025 10:28:19 by Admin

Updated 25 April 2025 10:13:17 by Admin