Einführung in die Programmierung

Aggregierte Datentypen

1. Aufzählungen

Aufzählungen werden benutzt, wenn eine Variable nur begrenzte Anzahl der Werte haben kann, z.B. Variable *AmpelFarbe* kann nur Werte *rot*, *gelb*, *grün* haben. Im Programm kann sie mit 0, 1, 2 kodiert werden, aber es ist schlecht lesbar — die Bezeichnungen *rot*, *gelb*, *grün* direkt im Source-Code wären besser.

Praktisch werden die Aufzählungen in Programmiersprachen als ganze Zahlen intern dargestellt, wobei die Werte mit 0 anfangen und aufsteigen, z.B. *rot=0*, *gelb=1*, *gruen=2*.

```
type Hochschule = (HWR, HTW, HBeuth);
var h : Hochschule;
BEGIN
  h := HTW;  // Kann man h von der Tastatur einlesen?
  WRITE(h);
  CASE h OF
  HWR:
    WRITELN(' 11.000 Studierende');
  HTW:
    WRITELN(' 13.000 Studierende');
  HBeuth:
    WRITELN(' 10.000 Studierende');
  END;
END.
```

```
type Hochschule int

const (
  HWR Hochschule = iota //  Inkrementierung
  HTW
  Hbeuth
)

func main() {
  var h Hochschule = HWR  // Kann man h von der Tastatur einlesen?
  fmt.Print(h)
  switch h {
  case HWR:
    fmt.Println(" 11.000 Studierende")
  case HTW:
    fmt.Println(" 13.000 Studierende")
  case Hbeuth:
    fmt.Println(" 10.000 Studierende")
  }
}
```

<u>Aufgabe 1.</u> Schreiben Sie Programme, wo Sie die arithmetischen Operatoren für Aufzählungen testen. Funktioniert die Eingabe von der Tastatur für Aufzählungen?

2. Reihungen

Eine Reihung (Array, Vektor) ist eine Zusammenfassung von mehreren Werten, die alle einen gemeinsamen Namen und gleichen Datentyp haben. Die einzelnen Werte (Elemente) werden durch Angabe des Namen und des Indexes angesprochen. Die Arrays haben immer Grenzen: Anfangswert (0 in vielen Programmiersprachen) und Endwert. Die Grenzen sind in vielen Sprachen statisch, d.h sie können im Laufe des Programms nicht verändert werden. Arrays mit dynamischen Grenzen gibt es auch.

```
VAR Datum : ARRAY [1..3] OF string;    // Deklaration
    Temperatur : ARRAY [1..3] OF real; // Anfangswert für Index kann auch anders sein
    i: 1..3;                    // Wäre eine Definition Datum : ARRAY [-7..4] gültig?
BEGIN
    FOR i := 1 TO 3 DO  // for-Schleife sollte für Verarbeitung
    BEGIN               // der Reihungen eingesetzt werden
        Datum[i] := '11.12.2020';
        Temperatur[i] := 12.8 + i;
    END;
    FOR i := 1 TO 3 DO
        WRITELN (Datum[i], Temperatur[i]:8:2);
END.
```

```
var Datum [3] string        // Deklaration
var Temperatur [3] float64  // Index nimmt Werte 0,1,2
for i := 0; i<3; i++ {      // i muss nicht deklariert werden
  Datum[i] = "11.12.2020"
  Temperatur[i] = 12.8 + float64(i)
}
for i := 0; i<3; i++ {
  fmt.Println(Datum[i], Temperatur[i])
}
```

<u>Aufgabe 2.</u> Schreiben Sie Programme, wo Sie testen, was passiert, wenn Index die Grenzen des Arrays überschreiten. Achten Sie auf die Fehlermeldungen.

<u>Aufgabe 3.</u> Schreiben Sie Programme, wo Sie die Eingabe von der Tastatur für die Elemente des Arrays testen. Lesen Sie von der Tastatur mehrere Fließkommazahlen ein, berechnen Sie den Mittelwert und geben Sie ihn aus, z.B. durchschnittliche Temperatur in einer Woche.

<u>Aufgabe 4.</u> Implementieren Sie folgendes: Definieren Sie zwei vierdimensionale Vektoren mit fließkomma-zahligen Komponenten — A (1.2, 2.3, 3.4, 4.5) und B (5.4, 4.3, 3.2, 2.1). Berechnen Sie deren Summe, Differenz und Skalarprodukt und geben Sie diese aus.

<u>Aufgabe 5.</u> Die Zeichenketten (Strings) sind in vielen Sprachen als Arrays von Symbolen (char, byte) definiert. Finden Sie heraus, wie man einzelne Symbole und sogar Teile aus einer Zeichenkette extrahieren kann. Lesen Sie eine Zeichenkette (8 bis 10 Symbole) von der Tastatur ein. Extrahieren Sie Symbol Nummer 0, 1, 2 aus. Dann extrahieren Sie eine Teilkette vom Symbol Nummer 3 bis Symbol Nummer 8.

<u>Aufgabe 6.</u> Implementieren Sie folgenden Algorithmus: Auf Ihrem Bildschirm sitzt ein Frosch mit einer Kamille im Maul. Das ist selbstverständlich ein Bild. Kamille hat die Koordinaten (20,6) auf dem Bildschirm. Der Frosch bewegt sich — der macht drei Sprünge, die durch Bewegungsvektoren (7,4), (3,5) und (8,4) angegeben sind. Berechnen Sie die entsprechenden drei Positionen der Kamille auf dem Bildschirm, und geben Sie sie aus.

3. Verbunde

Ein Verbund (Record, Struktur) ist eine Zusammenfassung von mehreren Werten (Elementen), die alle einen gemeinsamen Namen und unterschiedliche Datentypen haben. Die Elemente eines Verbundes haben unterschiedliche Namen, die durch einen Punkt den Verbundnamen ergänzen (qualifizieren). Verbund ist beim Transport der Daten sehr hilfreich, wenn man z.B. die Informationen über Objekte auf der Festplatte zu speichern sind.

```
TYPE Auto = RECORD
  KFZ:           String[12];   { 12 ist die max. Länge der Zeichenkette, optional }
  Gewicht:       Real;
  AnzPassagiere: Integer;
END;
VAR a : Auto;
BEGIN
  a.KFZ := 'B XY 123456';
  a.Gewicht := 555.77;
  a.AnzPassagiere := 5;
  WRITELN (a.KFZ, ' -- ', a.Gewicht:8:2, ' -- ', a.AnzPassagiere);
END;
```

```
type Auto struct {
  KFZ string
  Gewicht float64
  AnzPassagiere int
}
func main() {
  var a Auto
  a.KFZ = "B XY 123456"
  a.Gewicht = 555.77
  a.AnzPassagiere = 5
  fmt.Println (a.KFZ, " -- ", a.Gewicht, " -- ", a.AnzPassagiere)
}
```

<u>Aufgabe 7.</u> Deklarieren Sie einen Verbund "Artikel", dessen Komponente sind: Bezeichnung (String), Nettopreis (Fließkommazahl), Mehrwertsteuerbetrag (Fließkommazahl) und Bruttopreis (Fließkommazahl). Mehrwertsteuer ist immer noch 19% (das ist eine Konstante). Von der Tastatur werden Bezeichnung und Nettopreis eingegeben. Ausgegeben wird eine Zeile mit Überschriften und eine Zeile mit allen oben aufgelisteten Komponenten. Man nimmt an, dass Bruttopreis gleich Nettopreis plus Mehrwertsteuerbetrag ist.

<u>Aufgabe 8.</u> Ergänzen Sie die vorige Aufgabe 7: Bezeichnung des Artikels und dessen Nettopreis werden zuerst eingelesen und dann überprüft — die Bezeichnung darf nicht leer sein und der Nettopreis muss positiv sein. Nur in diesem Fall kommt die angeforderte Ausgabe, sonst kommt eine Meldung, die genau den Fehler beschreibt.

4. Mengen

Eine Menge (Set) kann mehrere Werte gleichzeitig haben. Zuerst wird eine Menge definiert, dabei muss festgelegt werden, welchen Datentyp die Elemente der Menge haben können. Dann kann man Variablen definieren, die einige (oder alle) Elemente beinhalten. Für Mengen ist Operator IN definiert (Pascal), oder wird ein logischer Ausdruck berechnet (Go).

```
TYPE Letters = SET OF CHAR;  // Gilt auch:  'a'..'z';
VAR Aus: Letters;            // Eine Variable !
  Eingabe: Char = 'a';
BEGIN
  Aus := ['q','Q','x','X'];  // Menü verlassen bei diesen Symbolen
  REPEAT                     // Menü anzeigen
    WRITELN('a – Append');
    WRITELN('d – Delete');
    WRITELN('x – Exit');
    WRITELN('Your choice > ');
    READLN (Eingabe);
  UNTIL (Eingabe IN Aus);    // Operator IN
END.
```

```
func main() {                    // byte = int8, rune = int32
  var j byte;                    // 'a' hat den Typ rune
  Menge := make(map[byte] bool)  // Variable Menge als map erstellen
  j = byte('a')                  // Umwandlung, oder überall byte durch rune ersetzen
  Menge[j] = true                // Zuordnung j zur Variable Menge
  fmt.Println ("-- ", Menge['a'], Menge['b'])
  delete(Menge, j)
  j = byte('b')                  // Ohne Zuordnung funktioniert nicht
  fmt.Println ("== ", Menge['a'], Menge['b'])
}

func main() {
  var Eingabe string;  // Einlesen von Tastatur ist besser, als byte - schlecht
  Aus := make(map[byte] bool)   // Variable Aus als map (Menge) erstellen
  Aus[byte('q')] = true         // Zuordnungen
  Aus[byte('Q')] = true
  Aus[byte('x')] = true
  Aus[byte('X')] = true
  for {                         // Menü
    fmt.Println("a – Append")
    fmt.Println("d – Delete")
    fmt.Println("x – Exit")
    fmt.Println("Your choice > ")
    fmt.Scan (&Eingabe)
    if Aus[Eingabe[0]] {        // 1. Element aus Eingabe
      break                     // Menü verlassen
    }
  }
}
```

Aufgabe 9. Sie arbeiten für Finanzamt in einem Schlaraffenland. Die Steuerklassen sind bei Ihnen nur 1, 4 und 6. Implementieren Sie folgenden Algorithmus: Name und Steuerklasse eines Kunden werden eingegeben. Steuerklasse ist der Einfachheit halber eine ganze Zahl. Verwenden Sie Menge, um zu überprüfen, ob eine gültige Steuerklasse eingegeben wurde. Entsprechende Meldung ist erwünscht.

5. Zusammenfassung

Aggregierte Datentypen können fast beliebig verschachtelt werden: man kann ein Array erstellen, wo Elemente die Verbunde sind, oder man kann einen Verbund erstellen, wo Elemente die Arrays sind.

Aufgabe 10. Ergänzen Sie die vorige Aufgabe 8: Definieren Sie ein vierdimensionales Array, wo jedes Element ein Artikel ist. Lesen Sie von der Tastatur Bezeichnung und Nettopreis für vier Artikel ein und speichern Sie diese Informationen im Array. Berechnen Sie die restlichen Komponenten und geben Sie eine Tabelle mit einer Überschrift und vier Zeilen aus. Nicht vergessen, die Eingabedaten auf Plausibilität zu überprüfen. Im Fehlerfall belegen Sie die Bezeichnung mit dem Wort "Fehler" und/oder den Nettopreis mit 0.

Aufgabe 11. Definieren Sie einen Verbund, der eine Woche folgenderweise beschreibt: Nummer der Woche, ein Array aus sieben Fließkommazahlen, die Temperatur an jedem Tag charakterisieren, und ein Array aus sieben Zeichenketten, die abgekürzte Bezeichnung des Wochentages enthalten ("Mo.", "Di.", u.s.w.). Definieren Sie ein Array mit 3 Komponenten, so dass jede Komponente einen oben beschriebenen Verbund darstellt. Lesen Sie alle Daten in dieses Array von der Tastatur ein und dann geben Sie sie auf dem Bildschirm aus.