Einführung in die Programmierung

Datentypen

1. Quellcode, Kommentare, das erste Programm

```
PROGRAM Mein1Programm;    (* Klein- und Großschreibung spielt keine Rolle *)
                          (* in Schlüsselwörtern, Variablen, Kontrollstrukturen *)
{$codepage utf8}          { Das ist auch ein Kommentar, }
                          { kann aber eine besondere Bedeutung wegen $ haben }

uses crt;                 { crt ist eine Standard-Bibliothek (Unit) }

BEGIN                     // Zeilenkommentar:
      CLRSCR;             // Standard-Programm zum Löschen des Bildschirms
      WRITE('It works !!!');    // ; ist hier überall wichtig
END.
```

```
package main       /* Klein- und Großschreibung ist wichtig */
import "fmt"       // fmt ist eine Standard-Bibliothek (Paket)

func main() {                        // ; ist wichtig, aber kommt sehr selten vor
      fmt.Printf("It works !!!\n")  // Printf() ist eine aus fmt sichtbare Funktion,
      fmt.Println("It works !!!")   // weil P großgeschrieben ist
}
```

2. Konstanten, Variable, Namen

Variable ist ein Bereich im Arbeitsspeicher, der einen Namen, einen Typ und einen Wert hat. Eine Variable wird durch ihren Namen angesprochen (verwendet). Sie kann im Laufe des Programms unterschiedliche Werte bekommen, aber zu einem Zeitpunkt nur einen.

Konstante ist ähnlich wie Variable aufgebaut, aber behält im Laufe des Programms immer nur einen Wert, der am Anfang des Programms einmal festgelegt wird.

Goldene Regel für Variablen- und Konstantennamen: Name beginnt mit einem lateinischen Buchstaben, weiter sind lateinische Buchstaben und Ziffern erlaubt (keine Leerzeichen, +, -, /, :, ä, ö, ß, u.s.w.).

Aufgabe 1. Schreiben Sie ein Programm in Pascal und in Go, wo Sie nur die Namen für Variablen und Konstanten testen: verwenden Sie die goldene Regel, und versuchen Sie auch "die anderen" Namen, die ein Leerzeichen enthalten, oder %, +, ä, ö, Unterstrich, oder die Namen, die mit einer Ziffer anfangen, wie 4u. Als Datentyp nehmen Sie etwas einfaches, wie integer. Ziel ist: Kompilation läuft ohne Fehler durch, und Ihr Programm startet.

3. Datentypen, Werte und Operationen

Typ einer Variable oder einer Konstante ist ganz wichtig. Davon ist abhängig, welche Operationen für den Wert der Variable erlaubt sind. Die Programmiersprachen, die ganz streng diesem Zusammenhang folgen, haben eine strenge Typisierung, z.B. Go und Pascal sind solche Sprachen, Python aber nicht. Die Zuordnung einer Variable zu einem Datentyp nennt man Deklaration. Unter Definition versteht man Deklaration und Zuweisung eines Wertes. Initialisie-

rung ist die Zuweisung des ersten Wertes in eine Variable.

Zuweisungsoperator (=, :=) funktioniert folgenderweise: Alles, was rechts von dem Zuweisungsoperator steht, wird zuerst berechnet (inlk. Funktionsaufrufe) und dann in die Variable, die links von dem Zuweisungsoperator steht, zugewiesen.

In Pascal muss man Zuweisungsoperator := von dem Initialisierungsoperator = unterscheiden. Der Zuweisungsoperator := wird in Blöcken (BEGIN ... END) und der Initialisierungsoperator = wird bei der Deklaration der Variablen (VAR) eingesetzt. Alle Variablen (und Konstanten) müssen in VAR definiert werden.

In Go werden dieselben Operatoren, aber in anderem Hintergrund, verwendet. Wurde eine Variable deklariert (var), so bekommt sie ihre Werte nur durch Operator =. Wurde eine Variable nicht deklariert, dann bekommt sie den ersten Wert mit dem Operator := und gleichzeitig bekommt den Datentyp des Ausdrucks, der rechts vom Zuweisungsoperator steht. Weitere Zuweisungen der Werte in diesem Fall passiert mit dem Operator =.

In Pascal darf man eine Variable nur einmal im VAR-Abschnitt zwischen PROGRAM (FUNCTION, PROCEDURE) und dem ersten BEGIN definieren, dagegen in Go kann man eine Variable (sogar mit dem selben Namen) mehrmals in unterschiedlichen { ... }-Blöcken definieren — das sind immer unterschiedliche Variablen.

```
PROGRAM Mein1Programm;

Var A : integer = 7;  //  Hier müssen alle Variablen definiert/deklariert werden

BEGIN
  Writeln (A);
  begin
    var A: integer = 42;  //  Syntaktischer Fehler. Darf man hier eine B definieren?
    Writeln (A);          //  Kommentieren Sie vorige Zeile, und starten Sie noch mal.
  end;
END.

Ausgabe:
7
7
```

```
package main
import "fmt"

var B int = 81         // Diese Variable ist überall, in allen Funktionen bekannt
func main() {
  var A int = 7
  fmt.Println(A,B)
  {                      //  Anfang des Blocks
    var A int = 42       //  Definition im Block – das ist eine neue Variable A !
    fmt.Println(A,B)     //  Sie ist nur in diesem Block bekannt
  }                      //  Ende des Blocks
  fmt.Println(A,B)
}

Ausgabe:
7 81
42 81
7 81
```

Atomare (einfache, nicht zusammengesetzte, direkt vom Compiler unterstützte) Datentypen in Pascal und in Go finden Sie z.B. hier

https://wiki.freepascal.org/Variables_and_Data_Types/de
https://www.codeflow.site/de/article/understanding-data-types-in-go

Aufgabe 2. Schreiben Sie ein Programm in Pascal und in Go, wo Sie nur Variablen und Konstanten definieren, aber von allen atomaren Typen, die Ihnen bekannt sind, wie Integer, Float- und Double-Zahlen, Zeichenketten, Boolean. Eigene Datentypen brauchen Sie noch nicht zu erstellen, genau so wenig brauchen Sie in dieser Aufgabe Ein- und Ausgabe, Berechnungen, Funktionen. Vielleicht verwenden Sie nur die Zuweisungen. Ziel ist: Kompilation läuft ohne Fehler durch, und Ihr Programm startet.

Operationen in Programmiersprachen sind meistens durch spezielle Symbole dargestellt, wie +, -, *, /, \, %, &, |, &&, || und durch die Schlüsselwörter wie mod, and, or, not. Eine Operation kann in mehreren Datentypen eingesetzt werden, selbstverständlich auch mit unterschiedlichen Bedeutungen.

Aufgabe 3. Schreiben Sie ein Programm in Pascal und in Go, wo Sie die Operationen für Integer und String vergleichen: + bedeutet Addition der Zahlen in Pascal und in Go, Konkatenation (Verketten) Pascal passiert mit +. gilt dasselbe für Go? Sie können Berechnungen mit oben genannten Operationen ausprobieren, einfache Ausgabe mit writeln(), fmt.Printf(), die Zuweisungen verwenden, sowie die Funktion length() für Zeichenketten. Mischen Sie unterschiedliche Datentypen in + Operation — ist es zulässig? Funktioniert length() für ganze Zahlen? Ziel ist: Kompilation läuft ohne Fehler durch, und Ihr Programm startet und zeigt die Ausgabe an. Merken Sie sich die Fehlermeldungen.

Aufgabe 4. Schreiben Sie ein Programm in Pascal und in Go, wo Sie die Initialisierungen bei der Deklaration der Variablen und die Zuweisungen der Werte vergleichen.

4. Formatierte Ausgabe

Die Werte, die in Variablen gespeichert sind, kann man auf passende Art und Weise bei der Ausgabe darstellen. Die Zahl 3,14159 kann so: 3,14, oder so: 3.14, oder so: 3, oder so: 03, dargestellt werden. Das ist wirklich nur die Darstellung — der Wert der Variable ändert sich dadurch nicht.

```
Name := 'Max Frei';       // In Pascal wird = niemals für Zuweisung verwendet, nur :=
Alter := 42;
Write(Name, ' ist ', Alter, ' Jahre alt', LineEnding);
// oder
WriteLn(Name, ' ist ', Alter, ' Jahre alt');

Pi := 3.14159;
WriteLn(Pi: 6:2);
```

```
Name := "Max Frei"        // Variable darf nicht vorher deklariert werden
Alter := 42               // Der Typ der Variable wird aus dem Wert ermittelt
                          // Das ist Unterschied zwischen := und = in Go

fmt.Println(Name, "ist", Alter, "Jahre alt")
// oder
fmt.Printf("%s ist %d Jahre alt", Name, Alter)

Pi := 3.14159; fmt.Printf("%6.4f", Pi); fmt.Printf("%.2f", Pi) // Hier ist ; wichtig
```

Aufgabe 5. Schreiben Sie Programme, wo Sie formatierte Ausgabe für ganze Zahlen und Fließkommazahlen, sowie für Zeichenketten testen.

5. Eingabe von Tastatur

Die Werte von Variablen werden oft von der Tastatur abgelesen. Schauen Sie sich die unten dargestellten Beispiele an. Finden Sie noch die anderen Möglichkeiten für Tastatureingabe in beiden Sprachen.

```
VAR i, j : INTEGER;
VAR s: String;
BEGIN
       I := 7;
       Write ('Geben Sie eine ganze Zahl ein --> '); Readln(i);
       Write ('.. und noch eine ganze Zahl --> '); Readln(j);
       Writeln(i,j); // Die Endung "ln" steht für "line"

       WRITE('Und jetzt eine Zeichenkette ');
       READLN(s); WRITELN(s);
END.
```

```
var s string
var i int
s = ""

fmt.Printf("\nGeben Sie eine Zahl ein ")
fmt.Scan(&i) // Testen Sie Wichtigkeit von &-Symbol im Eingabe-Operator Scan

fmt.Printf("\nGeben Sie eine Zeichenkette ein ")
fmt.Scan(&s) // &-Symbol stellt den Adress-Operator dar

fmt.Println("\nZahl ", i, ", String ", s); fmt.Println("EOT") // Hier ist ; wichtig
```

Aufgabe 6. Schreiben Sie Programme, wo Sie die Fließkommazahlen von der Tastatur eingeben. Sind bei der Eingabe , und/oder . möglich?

Aufgabe 7. Schreiben Sie Programme, wo Sie die ganzen Zahlen und die Fließkommazahlen von der Tastatur eingeben. Wie groß können die Zahlen sein (inkl. Nachkommastellen)?

Aufgabe 8. Schreiben Sie Programme, wo Sie einen Preis, den Sie in einem Geschäft bezahlt haben, eingeben. Nichts außer dem Preis soll eingegeben werden. Mehrwertsteuersatz beträgt 19%, und nämlich immer. Wenn man Ihr Programm mehrmals startet, bleibt Mehrwertsteuersatz derselbe. Berechnen Sie und geben Sie aus: Mehrwertsteuer, die der eingegebene Preis enthält, und die Differenz zwischen dem Preis und Mehrwertsteuer (Netto). Gehen Sie davon aus, dass Sie nur ein Produkt gekauft haben. Zum Testen — der bezahlte Preis 100,00 enthält 15,97 der MwSt und 84,03 von Netto.

Aufgabe 9. Recherchieren Sie, welche Informationen über sich selbst, über das Datum, über Version, über das Programm sollten Sie als Kommentar in jedem Programm schreiben.