Einführung in die Programmierung

Kontrollstrukturen (Steuerungskonstruktionen)

1. Block

Unter einer Anweisung versteht man Zuweisung (Ergebnis der Berechnung), Ein- oder Ausgabe-Operator, unären Operator, Aufruf einer Funktion oder einer Prozedur. Kontrollstrukturen in vielen Programmiersprachen erlauben nur eine Anweisung zu verwenden, aber oft sind an dieser Stelle mehrere Anweisungen notwendig. Als Lösung gilt hier ein Block. Ein Block kann mehrere Anweisungen enthalten, wird aber vom Compiler als eine einzige Anweisung betrachtet. Ein Block wird im Programm besonders gekennzeichnet (mittels Schlüsselwörtern oder speziellen Symbolen).

2. Einfache Auswahl

In der einfachen Auswahl gibt es eine Bedingung (ein Prädikat), einen Ja-Zweig (Ja-Block) und einen Nein-Zweig (Nein-Block). Bedingung muss logisch formuliert werden, d.h. sie muss entweder True oder False liefern. Der Ja-Zweig muss vorhanden sein, der Nein-Zweig kann fehlen. Einfache Auswahl kann verschachtelt werden, d.h. sowohl Ja-Zweig, als auch Nein-Zweig können weitere einfache Auswahl enthalten.

Die Bedingung in der einfachen Auswahl muss so formuliert werden, dass sie entweder "Ja" oder "Nein" liefert. Genau gesagt, diese Bedingung muss einen logischen (booleschen) Ausdruck darstellen.

In Pascal ist die Platzierung der Schlüsselwörter IF/THEN/ELSE recht freizügig, in Go ist die Platzierung der Symbole if/else/{/} im Gegenteil sehr streng. Die unten dargestellte Schreibweise widerspiegelt die Strukturierung der Programme, d.h. die Einbettung (Zugehörigkeit) der Blöcke ist durch deren Verschiebung gekennzeichnet.

```
ReadLn (ganzeZahl); // Eine Anweisung
IF ganzeZahl > 0    // Einfache Auswahl
THEN                // Ja-Zweig
BEGIN               // Block-Anfang
  WriteLn ('Dies ist Ja-Zweig');
  WriteLn ('Diese Zahl ist positiv');
END;                // Block-Ende

ReadLn (ganzeZahl); // Eine Anweisung
IF ganzeZahl > 0    // Einfache Auswahl
THEN                // Ja-Zweig
BEGIN               // Block-Anfang
  WriteLn ('Dies ist Ja-Zweig');
  WriteLn ('Diese Zahl ist positiv');
END                 // Block-Ende            (* Beachten Sie die ; *)
ELSE
BEGIN               // Block-Anfang
  WriteLn ('Dies ist Nein-Zweig');
  WriteLn ('Diese Zahl ist NICHT positiv');
END;                // Block-Ende
```

```
if ganzeZahl > 0 {   // Einfache Auswahl, Block-Anfang
  fmt.Println("Dies ist Ja-Zweig") // Ja-Zweig
  fmt.Println("Diese Zahl ist positiv")
}                    // Block-Ende

if ganzeZahl > 0 {   // Einfache Auswahl, Block-Anfang
  fmt.Println("Dies ist Ja-Zweig") // Ja-Zweig
  fmt.Println("Diese Zahl ist positiv")
} else {             // Block-Ende, Block-Anfang         /* Beachten Sie die ; */
  fmt.Println("Dies ist Nein-Zweig")
  fmt.Println("Diese Zahl ist NICHT positiv")
}                    // Block-Ende
```

Aufgabe 1. Schreiben Sie Programme in Pascal und in Go für die Unterscheidung der geraden/ungeraden Zahlen. Zuerst lesen Sie eine ganze Zahl ein, berechnen Sie den Rest der Division dieser Zahl durch 2, dann verwenden Sie die einfache Auswahl.

Aufgabe 2. Schreiben Sie Programme in Pascal und in Go für die richtige Erkennung des Alters. Alter des Kunden wird von der Tastatur eingegeben. Ist Alter kleiner als 12, dann ist es ein Kind. Ist Alter größer als 18, dann ist es ein Erwachsener. Liegt das Alter dazwischen, dann ist es ein Jugendlicher. Ihr Programm gibt diese Bezeichnungen "Kind", "Jugendlicher", "Erwachsener" auf dem Bildschirm aus. Wird eine negative Zahl oder Null als Alter eingegeben, dann muss entsprechende Fehlermeldung angezeigt werden. Verwenden Sie eingebettete einfache Auswahl. Achten Sie auf Strukturierung.

3. Mehrfache Auswahl

Einfache Auswahl genügt eigentlich, um alle Algorithmen, wo eine Auswahl notwendig ist, zu realisieren. Durch die Verschachtelung entsteht aber eine schwer nachvollziehbare Struktur. Deswegen ist es empfohlen, nur zwei bis drei Ebenen der eingebetteten einfachen Auswahl zu verwenden. Als Lösung in Situationen mit vielen Verschachtelungsebenen gilt die mehrfache Auswahl.

Mehrfache Auswahl hat keine logische Bedingung, sondern einen arithmetischen Ausdruck (Selector). Abhängig von der Programmiersprache sind auch die anderen Selectoren erlaubt, wie Zeichenketten, Aufzählungen.

```
Write ('Geben Sie Nummer des Fachbereiches ein: ');
ReadLn (ganzeZahl);
Write ('FB', ganzeZahl, ' - ');
CASE ganzeZahl OF
  1: WriteLn ('Wirtschaftswissenschaften');  // Eine Anweisung ist erlaubt
  2: WriteLn ('Duales Studium Wirtschaft Technik');
  3: WriteLn ('Allgemeine Verwaltung');
  4: WriteLn ('Rechtspflege');
  5: WriteLn ('Polizei und Sicherheitsmanagement');
ELSE
  WriteLn ('Unbekannt');  // Mehrere Anweisungen sind erlaubt
  WriteLn ('??????????');
END;
```

```
fmt.Print("Geben Sie Nummer des Fachbereiches ein:")
fmt.Scan(&ganzeZahl)
fmt.Print("FB", ganzeZahl, " - ")
switch ganzeZahl {
case 1:
  fmt.Println("Wirtschaftswissenschaften")
case 2:
  fmt.Println("Duales Studium")  // Mehrere Anweisungen sind erlaubt
  fmt.Println("Wirtschaft Technik")
case 3:
  fmt.Println("Allgemeine Verwaltung")
case 4:
  fmt.Println("Rechtspflege")
case 5:
  fmt.Println("Polizei und Sicherheitsmanagement")
default:
  fmt.Println("Unbekannt")  // Mehrere Anweisungen sind erlaubt
  fmt.Println("??????????")
}                            /* Achten Sie auf den Aufbau dieser Kontrollstruktur */
```

Aufgabe 3. Sowohl Pascal als auch Go erlauben auch die Zeichenketten als Selector. Recherchieren Sie, wie wird es gemacht.

Aufgabe 4. Schreiben Sie Programme, die als Eingabe die Kurzbezeichnung des Wochentages hat ("mo", "di","mi", u.s.w. Als Ausgabe kommt die volle Bezeichnung des Wochentages ("Montag", "Dienstag", "Mittwoch", u.s.w.). Bei einer falschen Eingabe muss entsprechende Fehlermeldung angezeigt werden. Verwenden Sie die mehrfache Auswahl.

Aufgabe 5. Schreiben Sie Programme, die Anzahl der Tage in einem Monat ausgeben. Monat wird als Zeichenkette eingegeben ("jan", "feb", "mrz", u.s.w.). Ausgegeben wird die entsprechende ganze Zahl. Schaltjahr sollen Sie erstmal nicht berücksichtigen, d.h. Februar hat immer 28 Tage. Verwenden Sie die mehrfache Auswahl.

Aufgabe 6. Man betrachte das vereinfachte Rechtesystem des Betriebssystems MS Windows. Es gebe drei Benutzergruppen: Admins, PowerUsers, Users und Guests, und zwei Arten von Rechten: Datei- und Druckerrechte. Diese Rechte seien von Benutzergruppen folgenderweise abhängig:

Benutzergruppe Admins
Dateirechte: Erstellen, Lesen, Löschen, Korrigieren, Rechte vergeben
Druckerrechte: Drucken, Aufträge verwalten, Rechte vergeben
Benutzergruppe PowerUsers
Dateirechte: Erstellen, Lesen, Löschen, Korrigieren
Druckerrechte: Drucken, Aufträge verwalten
Benutzergruppe Users
Dateirechte: Erstellen, Lesen, Korrigieren
Druckerrechte: Drucken
Benutzergruppe Guests
Dateirechte: Lesen
Druckerrechte: Drucken

Ihr Programm liest von der Tastatur folgende Angaben ein: den Anmeldename (Konto) eines Benutzers, zu welcher Gruppe gehört dieses Konto und die Art der Rechten — ein F für Dateirechte, ein P für Druckerrechte. Als Ausgabe kommt Anmeldename, Gruppe und Auflistung der Rechte, ähnlich wie oben dargestellt. Setzen Sie einfache Auswahl innerhalb einer mehrfachen Auswahl ein. Achten Sie auf strukturierte Schreibweise.

4. Anforderungen an Strukturierung in Schleifen

Die Prinzipien der Strukturierung widerspiegeln sind im Aufbau der Schleifen, nämlich, eine Schleife muss genau einen Eingang und genau einen Ausgang haben. Somit muss eine Schleife durch die Bedingung verlassen werden, nicht durch Sprung-Operatoren (continue, break, return) im Körper der Schleife. Da ist die Theorie, die Praxis sieht ein wenig anders aus. Viele Programmiersprachen lassen die Sprung-Operatoren im Körper der Schleife zu. Versuchen Sie erstmal in diesen Übungen der Theorie zu folgen.

5. Kopfgesteuerte Schleife

Bedingung der kopfgesteuerten Schleife wird immer vor der nächsten Ausführung des Körpers ausgewertet. Kopfgesteuerte Schleife wird u.U. niemals ausgeführt, nämlich wenn die Bedingung schon bei erster Auswertung falsch ist. Genau in solchen Situationen ist die kopfgesteuerte Schleife empfehlenswert.

```
VAR ganzeZahl : integer;
BEGIN
  ganzeZahl := -1;
  WHILE ganzeZahl <= 0 DO // Solange die Zahl nicht positiv ist, läuft die Schleife
  BEGIN                   // Typische kopfgesteuerte (while-) Schleife
    WRITELN('Eine positive Zahl eingeben ');
    READLN(ganzeZahl);
  END;
  WRITELN('Ja... endlich...');
END.

(****  Ewige Schleife ****)
WHILE 0 <= 1 DO
BEGIN
  WRITELN('Eine positive Zahl eingeben ');
  READLN(ganzeZahl);
END;
```

```
func main() {   // for-Schleife implementiert in Go alle Arten von Schleifen
  var ganzeZahl int
  ganzeZahl = -1
  for ganzeZahl <= 0 { // Solange die Zahl nicht positiv ist, läuft die Schleife
    fmt.Println("Eine positive Zahl eingeben ")
    fmt.Scan(&ganzeZahl)
  }
  fmt.Println("Ja... endlich...")
}

for {   /****  Ewige Schleife ****/
  fmt.Println("Eine positive Zahl eingeben ")
  fmt.Scan(&ganzeZahl)
}
```

<u>Aufgabe 7.</u> Implementieren Sie folgenden Algorithmus für die Ermittlung des größten gemeinsamen Teilers von zwei ganzen Zahlen (ggT). Zwei ganze Zahlen werden von der Tastatur eingelesen. Beide müssen positiv sein, sonst kommt eine Fehlermeldung. Verwenden Sie dafür einfache Auswahl. Solange die Zahlen nicht gleich sind, wird die größte durch eine Differenz ersetzt (größte − kleinste). Verwenden Sie dafür kopfgesteuerte Schleife. Sind die Zahlen gleich, so stellen sie den gesuchten ggT dar.

6. Zählschleife

Die Zählschleife ist eine Art der kopfgesteuerten Schleifen, d.h. die Bedingung wird jedes Mal vor der Ausführung des Körpers überprüft. Bei der Zählschleife werden die Grenzen der Ausführung festgelegt: Steuerungsvariable der Schleifen läuft exakt vom Anfangswert bis zum Endwert (inkl.). Die Zählschleife passt sehr gut für Verarbeitung von Reihungen (Arrays). Für diese Zwecke existieren weiter Varianten der Zählschleife.

```
{ Die Zahlen von 3 bis 7 addieren }
VAR Zahl : integer;
    Summe : integer;
BEGIN
  Summe := 0;
  FOR Zahl := 3 TO 7 DO
  BEGIN
    Summe += Zahl;
    // Dasselbe wie Summe = Summe + Zahl;
    // Operator ++ gibt es nicht
  END;
  WRITELN('Summe = ', Summe);
END.
```

```
/* Die Zahlen von 3 bis 7 addieren */
var Zahl int
var Summe int
Summe = 0
for Zahl = 3; Zahl <= 7; Zahl++ {
  Summe += Zahl
  // Dasselbe wie Summe = Summe + Zahl
  // Operator ++ gibt es
}
fmt.Println("Summe = ", Summe)
```

<u>Aufgabe 8.</u> Schreiben Sie Programme, wo Sie Fakultät einer von der Tastatur eingegebenen positiven Zahl berechnen und ausgeben. Wird eine negative Zahl oder Null eingegeben, kommt entsprechende Fehlermeldung.

<u>Aufgabe 9.</u> Implementieren Sie folgenden Algorithmus. Zwei ganze Zahlen werden von der Tastatur eingelesen. Die müssen unterschiedlich sein. Zuerst muss man erkennen, welche Zahl die kleinste ist. Alle ganzen Zahlen zwischen diesen zwei eingegebenen Zahlen werden untersucht, und diejenigen, die durch 7 teilbar sind, werden ausgegeben. Verwenden Sie dafür die Zählschleife, Anfang der Schleife gibt die kleinste Zahl, Ende — die größte.

7. Fußgesteuerte Schleife

Bedingung der fußgesteuerten Schleife wird immer nach der Ausführung des Körpers ausgewertet. Fußgesteuerte Schleife wird immer mindestens einmal ausgeführt, unabhängig von der Bedingung.

Theoretisch wird der Körper der Schleife nur dann ausgeführt, wenn die Bedingung TRUE ist. Das betrifft die Schleifen jeder Art. Praktisch, stellen die Programmiersprachen die Konstruktionen bereit, wo die Ausführung stattfindet, wenn die Bedingung FALSE ist (fußgesteuerte Schleife in Pascal).

Fußgesteuerte Schleifen sind in den Situationen nützlich, wo eine Aktion mindestens einmal durchgeführt werden soll, z.B. Ausgabe eines Menüs.

```
VAR Zahl : integer;
BEGIN
  REPEAT
    Write ('Geben Sie eine Zahl ein, oder 0 zum verlassen ');
    ReadLn (Zahl);
    WRITELN('Quadrat von ', Zahl, ' ist ', Zahl * Zahl);
  UNTIL Zahl = 0;
END.
```

```
func main() {
  var Zahl int
  for {      // Ewige Schleife
    fmt.Println ("Geben Sie eine Zahl ein, oder 0 zum verlassen ");
    fmt.Scan (&Zahl);
    fmt.Println ("Quadrat von ", Zahl, " ist ", Zahl * Zahl);
    if Zahl == 0 {
      break  // Das ist Verletzung der Strukturierung,
    }        // aber anders geht es in Go nicht
  }
}
```

Aufgabe 10. Schreiben Sie Programme, wo Sie ein Menü eines Restaurants auf dem Bildschirm anzeigen. Menü besteht aus mehreren Punkte, der letzte Punkt steht für Verlassen des Menüs (und somit auch des Programms). Benutzer gibt auf der Tastatur die Nummer des Menüpunktes ein. Solange nicht der letzte Punkt eingegeben wird, kommt eine kurze Beschreibung des ausgewählten (eingegebenen) Punktes.

Aufgabe 11. Quadratische Gleichung lösen.

Aufgabe 12. Horizont eines Planeten berechnen.