Aufgabe 1: Sortierverfahren

Sortieren Sie die folgende Liste von Zahlen aufsteigend mit dem Verfahren *Insertion Sort*:

38 18 5 21 29 14 35

Geben Sie die Liste nach jedem Durchlauf der inneren Schleife an, d.h. nach jedem vollständigen Einsortieren eines Elements.

Lösung

Die folgenden Zwischenergebnisse entstehen bei einem In-Place durchgeführten Insertion Sort. Sortierter und unsortierter Teil der Liste sind jeweils durch einen senkrechten Strich (|) getrennt:

38	|	18	5	21	29	14	35
18	38	|	5	21	29	14	35
5	18	38	|	21	29	14	35
5	18	21	38	|	29	14	35
5	18	21	29	38	|	14	35
5	14	18	21	29	38	|	35
5	14	18	21	29	35	38	|

Aufgabe 2: Syntax von Programmen

Welche der folgenden Variablendeklarationen sind syntaktisch korrekt? Bei den nicht korrekten Zeilen erläutern Sie jeweils kurz den Fehler.

```
1   x := 42
2   var int y 55
3   int z = 42
4   s := string([]byte{'a', 'b', 'c'})
5   b := []byte{'a', 'b', 'c'}
6   var l1 []int := make([]int, 0)
7   string := hallo
```

Lösung

Die folgenden Zeilen sind nicht korrekt:

- Zeile 2: Typ und Name der Variablen sind vertauscht und es fehlt ein Gleichheitszeichen.
- Zeile 3: Typ und Name der Variablen sind vertauscht und es fehlt das `var` oder statt der Typangabe ein `:=`
- Zeile 6: Die `var`-Form kann nicht mit der `:=`-Form kombiniert werden.
- Zeile 7: Bei `hallo` fehlen Anführungszeichen. (*Anmerkung:* Das Schlüsselwort `string` sollte zwar besser nicht als Name verwendet werden, aber dies ist kein Syntaxfehler.)

Aufgabe 3: Fehlersuche: Syntaxfehler

Das folgende Progamm enthält eine Reihe an Syntaxfehlern, durch die es nicht compiliert. Markieren Sie alle Zeilen, die einen Fehler enthalten und erläutern Sie kurz, was jeweils falsch ist.

```
1  package foo
2
3  import "fmt"
4
5  func PrintSomething(what string) string {
6    fmt.Print(what)
7    fmt.Print("\n")
8  }
9
10 func ComputeProduct(numbers int) int {
11   result := 1
12   for _, num := range numbers {
13     result *= num
14   }
15   return result
16 }
17
18 func main() {
19   p = ComputeProduct(1, 3, 5, 2, 0, 2)
20   PrintSomething(string fmt.Sprint(p))
21 }
```

Hinweis: Es geht hier nicht um inhaltliche Fehler, nur um Syntaxfehler.
Anmerkung: Für jede falsch markierte Zeile gibt es Punktabzug!

Lösung

Im Programm sind folgende Fehler:

- Zeile 1: Das Package muss `main` heißen, weil es eine `main`-Funktion gibt.
- Zeile 5: Funktion hat einen Return-Typ, aber kein `return`.
- Zeile 10/12: numbers ist `int`, unten in der Schleife wird aber darüber iteriert.
- Zeile 19: Die neue Variable darf nicht mit `=` definiert werden, sondern mit `:=`.
- Zeile 20: Beim Aufruf der Funktion darf der Typ des Arguments (hier `string`) nicht mit angegeben werden.

Aufgabe 4: Fehlersuche: Inhaltliche Fehler

Die folgende Funktion ist zwar syntaktisch korrekt, sie erfüllt aber nicht ihre Aufgabe. Erläutern Sie den/die Fehler und machen Sie einen Vorschlag zur Korrektur.

```
1  // IsPrime liefert true, falls n eine Primzahl ist.
2  func IsPrime(n int) bool {
3    for i := 2; i < n-1; i++ {
4      if n%i == 0 {
5        return false
6      } else {
7        return true
8      }
9    }
10   return true
11 }
```

Anmerkung: Ihr Korrekturvorschlag muss kein syntaktisch korrekter Code sein. Eine Erklärung in Worten genügt.

Lösung

Es gibt zwei Fehler in diesem Code:

1. Der Fall $n \leq 1$ wird nicht abgefangen. Dadurch wird z.B. $n = 1$ fälschlicherweise als Primzahl erkannt. Dies kann mit einer zusätzlichen Abfrage vor der Schleife behoben werden.

2. Der `else`-Fall in der Schleife ist falsch. Das `else` führt dazu, dass die Schleife sofort im ersten Durchlauf abbricht und dadurch jede Zahl als Primzahl erkennt, die nicht durch 2 teilbar ist. Das `else` kann in diesem Fall einfach weggelassen werden.

Eine korrekte Version der Funktion wäre z.B. die folgende:

```
// IsPrime liefert true, falls n eine Primzahl ist.
func IsPrime(n int) bool {
  if n <= 1 {
    return false
  }
  for i := 2; i < n-1; i++ {
    if n%i == 0 {
      return false
    }
  }
  return true
}
```

Aufgabe 5: Programmverständnis

Erläutern Sie, was die folgende Funktion berechnet. Geben Sie eine möglichst allgemeine bzw. abstrakte Erklärung an. Erklären Sie auch, mit welcher Art von Argumenten diese Funktion sinnvoll arbeitet.

```
1 func Foo(m, n int) bool {
2   if m == 0 || m > n {
3     return false
4   }
5   if n == m {
6     return true
7   }
8   return Foo(m, n-m)
9 }
```

Lösung

Die Funktion berechnet, ob m ein Teiler von n ist.
Folgende Fälle können eintreten:

- Wenn $m = 0$ oder $m > n$ gilt, kann n nicht durch m teilbar sein.
- Wenn die beiden Zahlen gleich sind, ist m ein Teiler.
- Wenn $m < n$, wird durch die Rekursion so lange m von n subtrahiert, bis m entweder kleiner oder gleich n ist. Der Rest bei der Division der beiden Zahlen ändert sich durch diese Subtraktionen nicht. Deshalb ist das Ergebnis `Foo(m,n-m)` auch für `Foo(m,n)` korrekt.

Aufgabe 6: Datenstrukturen

Entwerfen Sie eine Datenstruktur, die geeignet ist um ein Textdokument zu verwalten, wie es z.B. in Textverarbeitungen wie Microsoft Word vorkommt.
Die Struktur soll Zugriff auf folgende Metadaten bzw. Dokumentteile bieten:

- Den gesamten Text.
- Die Anzahl der Wörter im Text.
- Für jede Stelle im Text die Schriftart.
- Datum der letzten Änderung.
- Autor der letzten Änderung.
- Alle am Dokument beteiligten Autoren

Begründen Sie die Wahl Ihrer Datentypen.
Anmerkung: Sie können auch weitere Hilfs-Datentypen oder Methoden definieren. Es ist nicht notwendig, syntaktisch korrekten Code zu schreiben.

Lösung

Eine mögliche Lösung wäre die folgende Sammlung von Structs:

```
type Date struct {
  Day, Month, Year int
}

type FontInfo struct {
  Begin, End int
  Fontname   string
}

type Document struct {
  Text         string
  WordCount    int
  DefaultFont  string
  Fontinfo     []FontInfo
  LastEditDate Date
  LastAuthor   string
  Authors      []string
}
```

Hier werden jeweils Datentypen für Daten, die Schriftart-Info und das eigentliche

Dokument definiert:

- Ein Datum besteht aus Zahlen für Tag, Monat und Jahr. Dadurch ist ein strukturierter Zugriff darauf möglich.
- `FontInfo` ist ein Struct, das einem Textabschnitt eine Schriftart zuordnet. Das Struct enthält die Position des Anfangs und des Endes des Abschnitts sowie den Namen der Schriftart als String.
- In `Document` wird dann eine Liste solcher FontInfos sowie eine Standardschriftart für alle übrigen Abschnitte gespeichert.