

Theorie-Programmieraufgaben

10. Oktober 2024

Aufgabe: Ausgabe von Programmen

Was gibt diese Funktion auf die Konsole aus?

```
1 func Foo1() {  
2     x := 8  
3     y := 7  
4     k := x + 2*y - 2  
5  
6     fmt.Println(x, y, k)  
7     fmt.Printf("%d-%d-%d\n", x, y, k)  
8     fmt.Println(k + 2 - x)  
9     fmt.Println(float64(k) / float64(x))  
10 }
```

Lösung

```
1 // Output:  
2 // 8 7 20  
3 // 8-7-20  
4 // 14  
5 // 2.5
```

Aufgabe: Ausgabe von Programmen

Was gibt diese Funktion auf die Konsole aus?

```
1 func Foo2(n int) {  
2     s := "*"   
3     for i := 0; i < n; i++ {  
4         fmt.Println(s)  
5         s += s  
6     }  
7 }
```

Lösung

```
1 // Output:  
2 // *  
3 // **  
4 // ****  
5 //  
6 // *  
7 // **  
8 // ****  
9 // ****  
10 //  
11 // *  
12 // **  
13 // ****  
14 // ****  
15 // ****
```

Aufgabe: Ausgabe von Programmen

Was gibt diese Funktion auf die Konsole aus?

```
1 func Foo3(n int) {  
2     if n == 0 {  
3         return  
4     }  
5     Foo3(n - 1)  
6     fmt.Println(strings.Repeat("*", n))  
7 }
```

Lösung

```
1 // Output:  
2 // *  
3 // **  
4 // ***  
5 //  
6 // *  
7 // **  
8 // ***  
9 // ****  
10 //  
11 // *  
12 // **  
13 // ***  
14 // ****  
15 // *****
```

Aufgabe: Ausgabe von Programmen

Was gibt diese Funktion auf die Konsole aus?

```
1 func Foo4(n int) {  
2     fmt.Print(n)  
3     n = 42  
4     fmt.Println(n)  
5 }
```

Lösung

```
1     // Output:  
2     // 342  
3     // 442  
4     // 542
```

Aufgabe: Ausgabe von Programmen

Was gibt diese Funktion auf die Konsole aus?

```
1 func Foo5() {  
2     n := 23  
3     Foo4(n)  
4     fmt.Println(n)  
5 }
```

Lösung

```
1 // Output:  
2 // 2342  
3 // 23
```

Aufgabe: Ausgabe von Programmen

Was gibt diese Funktion auf die Konsole aus?

```
1 func Foo6() {  
2     x := 1  
3     {  
4         x := 2  
5         {  
6             x := 3  
7             fmt.Println(x)  
8         }  
9         fmt.Println(x)  
10    }  
11    fmt.Println(x)  
12 }
```

Lösung

```
1 // Output:  
2 // 3  
3 // 2  
4 // 1
```

Aufgabe: Ausgabe von Programmen

Was gibt dieses Programm auf die Konsole aus?

```
1 func Foo() {
2     x := 3
3     y := 4
4     fmt.Println(x, y)
5     x = Bar(x, y)
6     fmt.Println(x, y)
7     Baz(x, y)
8     fmt.Println(x, y)
9 }
10
11 // Bar ist eine Hilfsfunktion für Foo.
12 func Bar(x, y int) int {
13     x = y - x
14     return x
15 }
16
17 // Baz ist eine Hilfsfunktion für Foo.
18 func Baz(x, y int) {
19     x, y = y, x
20     fmt.Println(x, y)
21 }
```

Lösung

```
1 // Output:
2 // 3 4
3 // 1 4
4 // 4 1
5 // 1 4
```

Aufgabe: Syntaxfehler

Welche Typen haben die Variablen `x`, `y`, `z` und `w`?

```
1      x := 42
2      y := x - 2
3      z := 3.14
4      w := "Hello"
```

Lösung

```
1      // Output:
2      // int int float64 string
```


Aufgabe: Syntaxfehler

Welche Typen haben die Variablen `x`, `y` und `z`?

```
1      x := Foo1()
2      y := Foo2()
3      z := Foo4(x)
4      z = z / 2.3
5      if y {
6          x += Foo3(z)
7      }
8      x += 3
```

Lösung

```
1      // Output:
2      // int bool float64
```

Aufgabe: Fehlersuche - Inhaltliche Fehler

Diese Funktion für einen Primzahltest ist fehlerhaft. Was ist falsch?

```
1 func IsPrimeBuggy(n int) bool {
2     for i := 2; i < n-1; i++ {
3         if n%i == 0 {
4             return false
5         } else {
6             return true
7         }
8     }
9     return true
10 }
```

Lösung

```
1 func IsPrimeCorrect(n int) bool {
2     /* Fehler 1: Spezialfall hat gefehlt, 0 und 1
       sind keine Primzahlen
3     if n <= 1 {
4         return false
5     }
6     // Im
7     for i := 2; i < n-1; i++ {
8         if n%i == 0 {
9             return false
10        }
11        /* Fehler 2: Der else-Zweig war falsch: Wenn
           n%i != 0,
12        /* dann ist n nicht automatisch prim, denn
           es kann immer
13        /* noch ein anderer Teiler gefunden werden.
14    }
15    return true
16 }
```

Aufgabe: Syntaxfehler

Finden Sie alle Fehler in den folgenden Variablendefinitionen.

```
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

```
1      x := 42
2      var y int = 55 /* var int y 55
3      var z int = 77 /* int z = 42
4      s := string([]byte{'a', 'b', 'c'})
5      b := []byte{'a', 'b', 'c'}
6      l := make([]int, 0) /* var l1 []int :=
        make([]int, 0)
7      string := "hallo" /* string := hallo
```

Aufgabe: Syntaxfehler

Finden Sie alle Fehler in diesem Programm.

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

Lösung

```
0 // Package muss main sein, weil es eine main-Funktion
  gibt.
1 package main
2
3 import "fmt"
4
5 // Funktion hat Return-Typ, aber kein return.
6 func PrintSomething(what string) {
7     fmt.Print(what)
```

```

8      fmt.Print("\n")
9  }
10
11 // numbers ist int, unten wird aber iteriert.
12 func ComputeProduct(numbers ...int) int {
13     result := 1
14     // (s.o.) Schleife mit int geht nicht.
15     for _, num := range numbers {
16         result *= num
17     }
18     return result
19 }
20
21 func main() {
22     // Neue Variable nicht mit = definieren.
23     p := ComputeProduct(1, 3, 5, 2, 0, 2)
24     // Typ nicht beim Aufruf mit angeben.
25     PrintSomething(fmt.Sprintf(p))
26 }

```

Aufgabe: Signaturen

Welche Signaturen haben die Funktionen Foo1, Foo2 etc.?

```
1      x1 := Foo1("Hallo", 15)
2      x2 := Foo2(x1)
3      x3 := Foo3(127, x2)
4      if x2 {
5          x3 = append(x3, Foo1("Welt", x1))
6      }
7      x1 += Foo4(x2, true)
8      Foo5(x2 && x1 != Foo4(x2, x2))
9      return x2 && !(x1 > x3[0])
```

Hinweis: Die Signatur einer Funktion gibt an, welche Parameter sie erwartet und welchen Typ der Rückgabewert hat.

Lösung

```
1      // Output:
2      // Foo1: func(string, int) int
3      // Foo2: func(int) bool
4      // Foo3: func(int, bool) []int
5      // Foo4: func(bool, bool) int
6      // Foo5: func(bool)
```

Aufgabe: Signaturen

Welche Signaturen haben die Funktionen Foo1, Foo2 etc.?

```
1      x1 := Foo1(42)
2      Foo2(x1)
3      s1 := []string{"Hallo"}
4      s2 := []string{"Welt"}
5      if Foo3(s1) {
6          x1 = Foo4(Foo3(s2), 15)
7      }
8      e1, e2 := Foo5(s1[0])
9      fmt.Printf("%d\n", x1+e1+e2)
```

Hinweis: Die Signatur einer Funktion gibt an, welche Parameter sie erwartet und welchen Typ der Rückgabewert hat.

Lösung

```
1      // Output:
2      // Foo1: func(int) int
3      // Foo2: func(int)
4      // Foo3: func([]string) bool
5      // Foo4: func(bool, int) int
6      // Foo5: func(string) (int, int)
```