IT-Handbuch für Fachinformatiker, 7. Auflage: Text-Lösungen

Sascha Kersken

Im Folgenden finden Sie jeweils die korrekte Antwort zu den im Buch abgedruckten Prüfungsfragen, soweit die Antworten in Textform gegeben werden sollen. Die Lösungen der reinen Programmieraufgaben können Sie dagegen als ZIP-Archiv herunterladen.

Lösungen zu Kapitel 2 (Mathematische und technische Grundlagen)

1. Rechnen Sie die Dezimalzahl 4.321 in die duale, oktale und hexadezimale Schreibweise um.

Dual: 1000011100001
Oktal: 10341
Hexadezimal: 10E1

2. Rechnen Sie die Dualzahl 11001100 in die dezimale, oktale und hexadezimale Schreibweise um.

Dezimal: 204
Oktal: 314
Hexadezimal: CC

3. Rechnen Sie die Oktalzahl 4567 in die dezimale, duale und hexadezimale Schreibweise um.

Dezimal: 2423
Dual: 100101110111
Hexadezimal: 977

4. Rechnen Sie die Hexadezimalzahl DCEF in die dezimale, duale und oktale Schreibweise um.

Dezimal: 56559
Dual: 1101110011101111
Oktal: 156357

5. Konzipieren Sie eine Turing-Maschine ...

Siehe grafische Darstellung und Erläuterungen auf der Website unter
http://buecher.lingoworld.de/fachinfo/aufgaben.html

6. Schreiben Sie die folgenden beiden Beispielprogramme für den virtuellen Prozessor:
- Zu dem Wert in Register A, hier n genannt, soll in Register B die Fakultät berechnet werden (n x x-1 x n-2 x ... x 1)

```
MOV A, 7  ; Beispielwert 7
MOV B, 1  ; Register B zurücksetzen/vorbereiten
LBL loop  ; Sprungmarke für Hauptschleife
CMP A, 1  ; A mit dem gewünschten Endwert 1 vergleichen
JBE end   ; Ende, wenn A <= 1
MUL B, A  ; B mit dem aktuellen Wert von A multiplizieren
DEC A     ; A um 1 vermindern
JMP loop  ; Unbedingter Sprung: Schleife
LBL end   ; Sprungmarke für Ende
HLT       ; Programm beenden
```

- Die Werte aus den Speicherstellen ab 0 bis zum Auftreten des Wertes 0 sollen in aufsteigend sortierter Reihenfolge auf den Stack gelegt werden

Diese Aufgabe ist eine Falle und soll Ihnen die Grenzen des virtuellen Prozessors vor Augen führen: da er keine indirekte Adressierung beherrscht (also eine Variable als Speicheradresse statt eines literalen Wertes), ist sie nicht lösbar!

Lösungen zu Kapitel 9 (Grundlagen der Programmierung)

1. Welche der folgenden Variablen sind gültige C-Bezeichner?

variable1 und **_variable** sind gültig. „1variable“ beginnt mit einer Ziffer statt einem Buchstaben oder einem Unterstrich, was verboten ist. „binärzahl“ enthält ein Nicht-ASCII-Zeichen („ä“). In vielen moderneren Sprachen wie Python oder Java wäre dieser Bezeichner gültig, in C jedoch nicht.

2. Finden Sie den Fehler im folgenden C-Programm, möglichst ohne es abzutippen und zu kompilieren

Die Variablen a und b werden in der Funktion main() lokal deklariert; die Funktion ergebnis() kann sie nicht kennen. Es gibt zwei Möglichkeiten, das Problem zu lösen: die Variablen global deklarieren oder sie als Übergabeparameter verwenden.

Lösung mit globaler Deklaration:

```
#include <stdio.h>

int a = 17;
int b = 29;

int ergebnis() {
  return a + b;
}

int main(int argc, char* argv[]) {
  printf("%d + %d = %d", a, b, ergebnis());
  return 0;
}
```

Lösung mit Übergabeparametern:

```
#include <stdio.h>

int ergebnis(int a, int b) {
  return a + b;
}

int main(int argc, char* argv[]) {
  int a = 17;
  int b = 29;
  printf("%d + %d = %d", a, b, ergebnis(a, b));
  return 0;
}
```

3. Wie prüfen Sie in C, ob die int-Variable a größer als 0 und kleiner als 10 ist?

a > 0 && a < 10

4. Welchen Wert hat die Variable ergebnis nach Ausführung der folgenden C-Operationen?

```
int a = 7;
int b = 9;
int ergebnis = ++a + b++;
```

Hier geht es um den Unterschied zwischen Prä- und Post-Inkrement. Die Variable a wird vor der Berechnung von ergebnis um 1 erhöht (Prä-Inkrement), hat also bereits den Wert 8, wenn ergebnis ermittelt wird. Dagegen wird b erst erhöht, nachdem ergebnis seinen Wert hat. Somit haben die

drei Variablen nach der Ausführung der drei Codezeilen folgende Werte:

```
a = 8
b = 10
ergebnis = 17  // 8 + 9
```

5. Der folgende C-Codeausschnitt soll von 10 bis 1 herunterzählen. Tut er dies? Falls nicht, was tut er stattdessen?

```
for (int i = 10; i > 1; i++) {
  printf("%d\n");
}
```

Der Code zählt ganz offensichtlich nicht herunter, denn die Wertänderungs-Anweisung lautet i++ (i um 1 erhöhen) und nicht i-- (vermindern). Ändert man dies, offenbart sich ein zweites Problem: es wird nur von 10 bis 2 gezählt, da die Wertüberprüfung i > 1 lautet. Beim Wert i == 1 wird der Schleifenrumpf also nicht erneut ausgeführt.

[Die Lösung zu Aufgabe 6 befindet sich im Download-ZIP]

7. Das folgende kleine Java-Programm soll die beiden Zahlen 23 und 42 addieren und das Ergebnis ausgeben. Tut es das? Falls nicht, was gibt es stattdessen aus?

```
public class Rechentest {
  public static void main(String[] args) {
    int a = 23;
    int b = 42;
    System.out.println("23 + 42 = " + a + b);
  }
}
```

Die Ausgabe lautet nicht, wie gewünscht, 65, sondern 2342. Das Problem ist, dass der Operator + sowohl der numerischen Addition als auch der String-Verkettung dient. Der ganz links stehende Operand ist ein String, dadurch werden auch die nachfolgenden Operanden als Strings interpretiert und die Operatoren entsprechend verwendet. Um die Aufgabe zu lösen, müsste das Ergebnis zuerst unabhängig von der Ausgabe berechnet und zwischengespeichert werden:

```
public class Rechentest {
  public static void main(String[] args) {
    int a = 23;
    int b = 42;
    int ergebnis = a + b;
    System.out.println("23 + 42 = " + ergebnis);
  }
}
```

8. Wie prüfen Sie in Java, ob die beiden Strings str1 und str2 denselben Inhalt haben?

str1.equals(str2)

9. Finden Sie alle Fehler in der folgenden Java-Klassendefinition

```
public class TestKlasse() {           // 1
  private int wert = "Hallo Welt";    // 2


  // Konstruktor
  public void TestKlasse(int _wert) {  // 3
    wert = _wert;
  }
```

```
  public int getWert() {
    return _wert;                       // 4
  }

  public int setWert(int _wert) {       // 5
    wert = _wert;
  }
}
```

Die Fehler sind rot markiert und durch die nummerierten Kommentare gekennzeichnet:

1. Hinter dem Klassennamen in einer Klassendefinition stehen keine runden Klammern.

2. Der Vorgabewert "Hallo Welt" ist ein String und passt nicht zum angegebenen Datentyp int.

3. Der Konstruktor wird ohne separate Datentypangabe geschrieben. Dieser Fehler wird nicht vom Compiler gefunden, denn es ist absolut legal, eine gewöhnliche Methode eines beliebigen Datentyps mit dem Namen der Klasse zu schreiben. Diese wird aber natürlich nicht aufgerufen, wenn eine Instanz der Klasse erzeugt wird.

4. Die lokale Variable _wert ist innerhalb der Methode getWert() nicht definiert; stattdessen müsste das Attribut wert zurückgegeben werden.

5. Die Setter-Methode wurde mit dem Datentyp int deklariert und müsste daher auch einen Wert dieses Typs zurückgeben. Dies tut sie aber nicht. Typischerweise geben Setter nichts zurück und erhalten den Datentyp void.

Hier die korrigierte Fassung der Klasse:

```
public class TestKlasse {
  private int wert = 0;

  // Konstruktor
  public TestKlasse(int _wert) {
    wert = _wert;
  }

  public int getWert() {
    return wert;
  }

  public void setWert(int _wert) {
    wert = _wert;
  }
}
```

[Die Lösung zu Aufgabe 10 befindet sich im Download-ZIP]

11. Welche der folgenden (in C gültigen) Ausdrücke sind auch in Python erlaubt?

Erlaubt sind nur `a & b` (bitweise Und) und `b -= 3` (b um 3 vermindern).

Das logische Und (in C: `a && b`) wird in Python als `a and b` geschrieben. Statt dem ternären Operator „`Bedingung ? Dann : Sonst`“ gibt es in Python das Konstrukt „`Dann if Bedingung else Sonst`“, im vorliegenden Fall also `b if a == 1 else c`. Die Inkrement- und Dekrement-Operatoren ++ und -- kennt Python gar nicht.

12. Was enthält die Python-Liste list nach Ausführung der folgenden Anweisungen?

```
list = [1, 2, 3, 4]
list[2:2] = [5, 6, 7]
```

Die Antwort lautet: `[1, 2, 5, 6, 7, 3, 4]`

Die Indexangabe `[2:2]` bedeutet, dass der Bereich ab Element Nummer 2 (das dritte Element, da ab 0 gezählt wird) *bis ausschließlich* Element Nummer 2 geändert wird – die neuen Werte werden also zwischen den Elementen Nummer 2 und 3 eingefügt; es wird nichts ersetzt.

[Die Lösungen zu Aufgabe 13 und 14 befinden sich im Download-ZIP]

15. Was gibt das folgende Python-Skript aus?

```
class A:
  def ausgabe(self):
    print("Ich bin A.")

class B(A):
  def ausgabe(self):
    print("Ich bin B.")

class C(B, A):
  pass

c = C()
c.ausgabe()
```

Die Ausgabe lautet „Ich bin B.“ – Die Klasse C erbt von den Klassen B und A (in dieser Reihenfolge), und bei gleichnamigen Methoden in Elternklassen „gewinnt“ die zuerst übernommene. Da die Klasse B ihrerseits von A abgeleitet wird, ist die hier vorliegende Deklaration in der Praxis übrigens vollkommen überflüssig – C erbt über B ohnehin diejenigen Methoden A, die B nicht überschreibt. Anders sähe es mit

```
class C(A, B)
```

aus – in diesem Fall würde C nur Methoden aus B übernehmen, die in A nicht implementiert sind.

[Die Lösung zu Aufgabe 16 befindet sich im Download-ZIP]