

http://ovm-kassel.de Lernjob	
Lernjob AE-MS-LJ-1.4 Pseudocode und Programmablaufpläne	
Code	AE-MS-LJ-1.4
Autor	André Bauer <a(dot)bauer(at)ovm-kassel(dot)de>
Datum	10. September 2018
Links	<ul style="list-style-type: none"> • code2flow • PapDesigner
Verwandte Lernjobs	AE-MS-LJ-1.1 bis 1.3
Lizenz	 Dieses Werk ist lizenziert unter einer Creative Commons Namensnennung - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz .

Pseudocode und Programmablaufpläne

1. Lineare Strukturen

1.1. Beispiel 1: Umrechnung von Pferdestärken in Kilowatt

Bevor die Einheit Watt zu Angabe der Leistung von Maschinen im amtlichen und geschäftlichen Bereich vorgeschrieben wurde, wurde insb. für Kraftfahrzeuge die Einheit Pferdestärke (PS) verwendet, die dort auch heute noch gebräuchlich ist. Das Programm soll eine Angabe in Pferdestärken in Kilowatt (kW) umrechnen und ausgeben.

$$1 \text{ PS} = 735,498 \text{ W}$$

Pseudocode

Quellcode 1. Pseudocode zur Berechnung der Leistung in kW

```

program LeistungVonPSinKW
  read ps;
  w := ps * 735.498;
  kw := w / 1000;
  print kw;
end programm LeistungVonPSinKW

```

Programmablaufplan

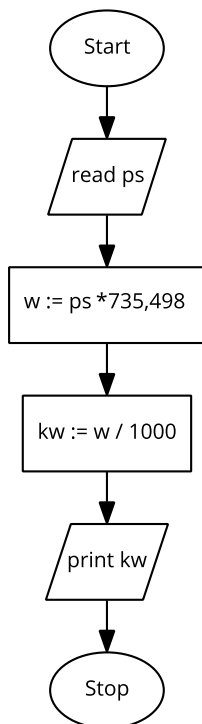


Abbildung 1. Programmablaufplan zur Berechnung der Leistung in kW

1.2. Aufgabe 1: Fahrradcomputer

Auf dem Display eines Fahrradcomputers sollen die seit dem letzten Zurücksetzen des Zählers zurückgelegte Strecke sowie die aktuelle Geschwindigkeit angezeigt werden.

Als Berechnungsgrundlage dient jeweils der Reifenaußendurchmesser in Zoll sowie die Anzahl der Umdrehungen. Für die zurückgelegte Strecke wird die Anzahl seit dem letzten Zurücksetzen des Zählers gemessen, für die Geschwindigkeit die Anzahl der Umdrehungen in den letzten drei Sekunden.

d = Reifenaußendurchmesser

s = zurückgelegte Strecke

v = momentane Geschwindigkeit

$$s = d \cdot 2,54 \frac{\text{cm}}{\text{Zoll}} \cdot \pi \cdot \text{Umdrehungen} \cdot \frac{\text{m}}{100 \text{ cm}}$$

$$v = d \cdot 2,54 \frac{\text{cm}}{\text{Zoll}} \cdot \pi \cdot \text{Umdrehungen} \cdot \frac{1}{3 \text{ s}} \cdot 20 \frac{\text{s}}{\text{min}} \cdot 60 \frac{\text{min}}{\text{h}} \cdot \frac{\text{m}}{100 \text{ cm}} \cdot \frac{\text{km}}{1000 \text{ m}}$$

2. Mehrfachauswahl

2.1. Beispiel 2: Römische Zahlschrift

In die römischen Zahlschrift werden üblicherweise sieben Buchstaben verwendet, die jeweils einem bestimmten Wert entsprechen.

Wert	Zeichen
1	I

Wert	Zeichen
5	V
10	X
50	L
100	C
500	D
1000	M

Das folgende Programm gibt zu einem Wert das entsprechende Zeichen zurück. Falls es für einen Wert kein Zeichen gibt, wird eine Fehlermeldung zurückgegeben.

Pseudocode

Quellcode 2. Pseudocode zur Funktion `roemischesZahlzeichen`

```
function roemischesZahlzeichen(wert)
  switch wert
    case 1:
      return I;
      break;
    case 5:
      return V;
      break;
    case 10:
      return X;
      break;
    case 50:
      return L;
      break;
    case 100:
      return C;
      break;
    case 500:
      return D;
      break;
    case 1000:
      return M;
      break;
    default:
      error: "Für den Wert " wert " gibt es kein römisches Zahlzeichen"
```

Programmablaufplan

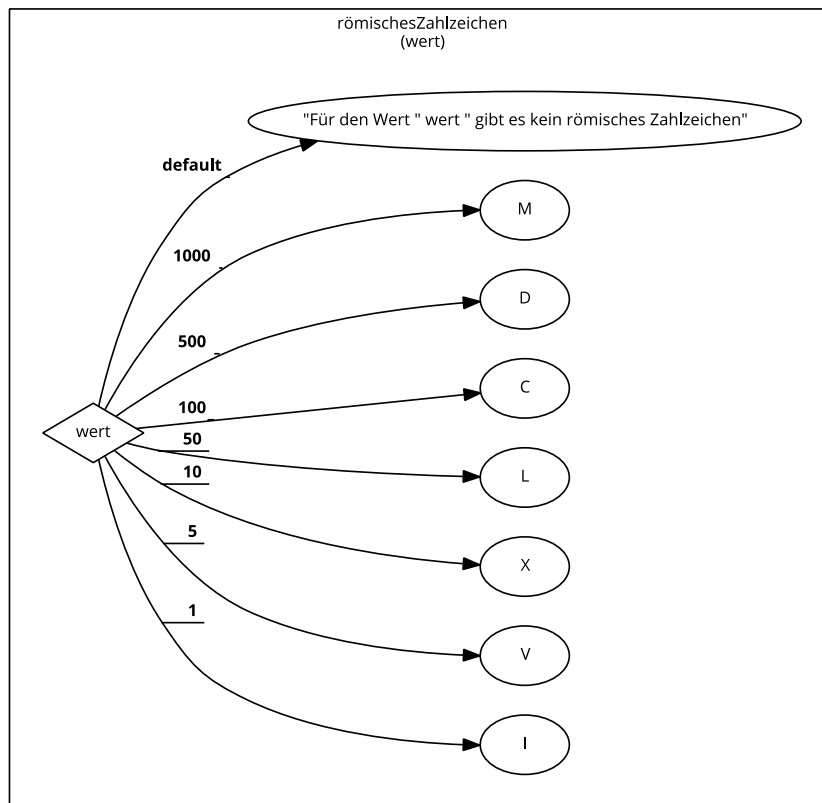


Abbildung 2. Programmablaufplan zur Funktion `römischesZahlzeichen`

2.2. Aufgabe 2: TCP-Netzwerkdienste

Ein Programm soll für eine Auswahl beliebiger TCP-Netzwerkdienste zu der Standard-Portnummer den entsprechenden Dienstnamen ausgeben. Falls die Portnummer unbekannt ist, soll eine entsprechende Fehlermeldung ausgegeben werden.

Standard-Portnummer	Dienstname
20	ftp
22	ssh
25	smtp
53	domain
80	http
110	pop3
443	https

3. Mehrfache Verzweigung

3.1. Beispiel 3 IPv4-Netzklassen

Die IPv4-Adressen haben eine Länge 32-Bit, d. h. vier Byte. Zur menschenlesbaren Darstellung ist es

üblich, die vier Bytes jeweils im Dezimalsystem, also als Zahl zwischen 0 und 255 darzustellen, und mit einem Punkt zu trennen. So hat die Domain ovm-kassel.de die IP-Adresse 89.31.143.1. Anhand der Anfangsbits bzw. der ersten Bytes wurden die IPv4-Adressen bis 1993 in sogenannte Netzwerkklassen unterteilt. Anhand der Dezimaldarstellung des ersten Bytes ergibt sich die folgende Tabelle für die Netzwerkklassen:

Bereich	Netzwerkklasse
0 – 127	A
128 – 191	B
192 – 223	C
224 – 239	D
240 – 255	E

Die IP-Adresse 89.31.143.1 liegt somit in der Netzwerkklasse A.

Pseudocode

Quellcode 3. Pseudocode zur Bestimmung der Netzwerkklasse

```
program Netzwerkklasse
  read firstByte;
  if firstByte < 0 oder firstByte > 255
    error "Ungültige Eingabe";
  if firstByte <= 127
    print "Klasse A";
  elseif firstByte <= 191
    print "Klasse B";
  elseif firstByte <= 223
    print "Klasse C";
  elseif firstByte <= 239
    print "Klasse D";
  else
    print "Klasse E";
end program Netzwerkklasse
```

Programmablaufplan

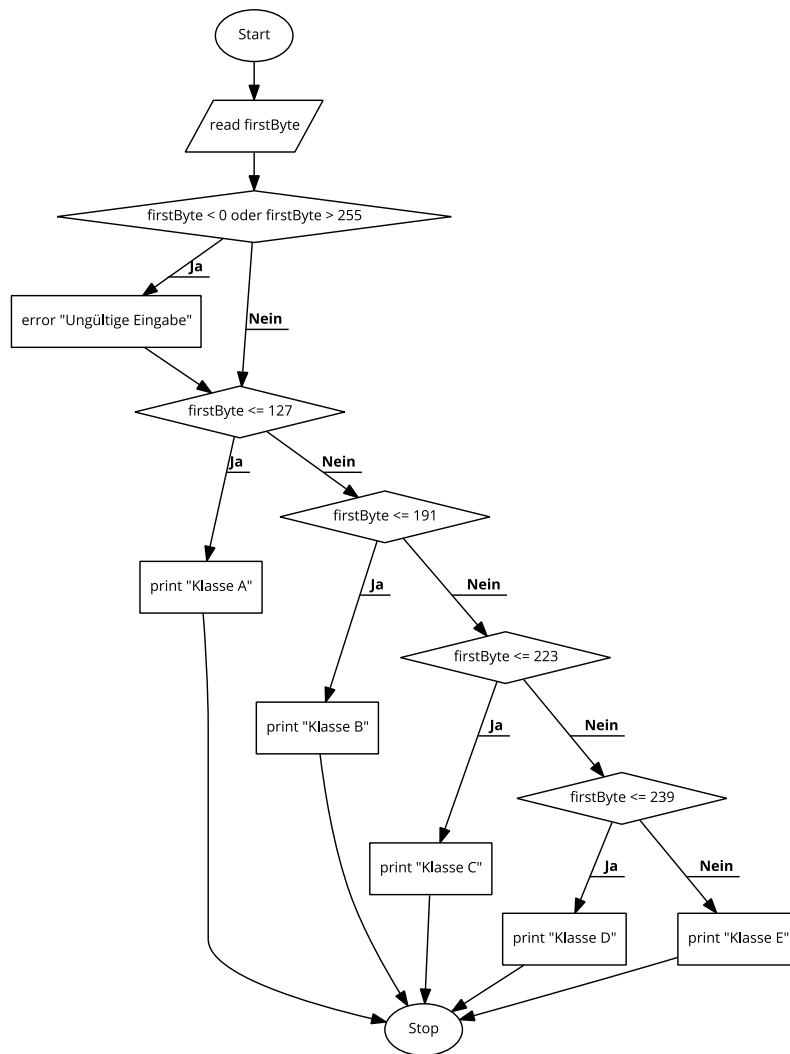


Abbildung 3. Programmablaufplan zur Bestimmung der Netzwerkklassse

3.2. Aufgabe 3: Helligkeits-Schwellwerte

Eine Mustererkennungssoftware benötigt zu jedem Bildpunkt eine Klassifizierung anhand der Helligkeit. Die Helligkeit eines Bildpunktes wird als 10-Bit-Wert, d. h. als Dezimalzahl zwischen 0 und 1023 gemessen. Die Funktion soll zu der Helligkeit den entsprechenden Farbnamen ausgeben.

Helligkeits-Bereich	Farbname
0 – 149	schwarz
150 – 400	dunkelgrau
401 – 624	grau
625 – 874	hellgrau
875 – 1023	weiß

4. Anfangsgeprüfte Schleife und Verzweigung

4.1. Beispiel 4: Primfaktorzerlegung

Jede natürliche Zahl kann als Produkt von Primzahlen geschrieben werden. Beispiele:

- $6 = 2 \cdot 3$
- $24 = 2 \cdot 2 \cdot 2 \cdot 3$
- $99 = 3 \cdot 3 \cdot 11$
- $1001 = 7 \cdot 11 \cdot 13$

In einem Programm sollen zu einer gegebenen natürlichen Zahl die Primfaktoren berechnet und ausgegeben werden.

Struktogramm

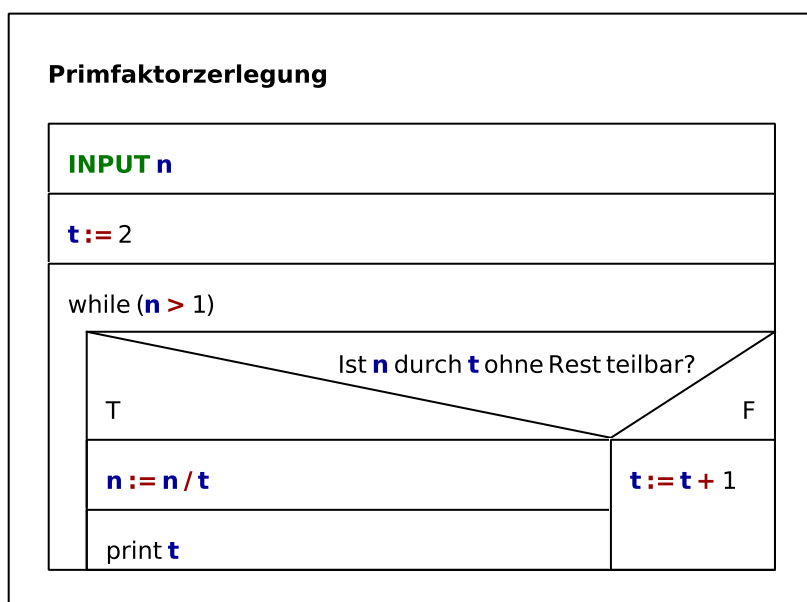


Abbildung 4. Struktogramm zur Primfaktorzerlegung

Pseudocode

Quellcode 4. Pseudocode zur Primfaktorzerlegung

```

program Primfaktorzerlegung
  read n;
  t := 2;
  while n > 1
    if Ist n durch t ohne Rest teilbar?
      n := n / t;
      print t;
    else
      t := t + 1
  end program Primfaktorzerlegung

```

Programmablaufplan

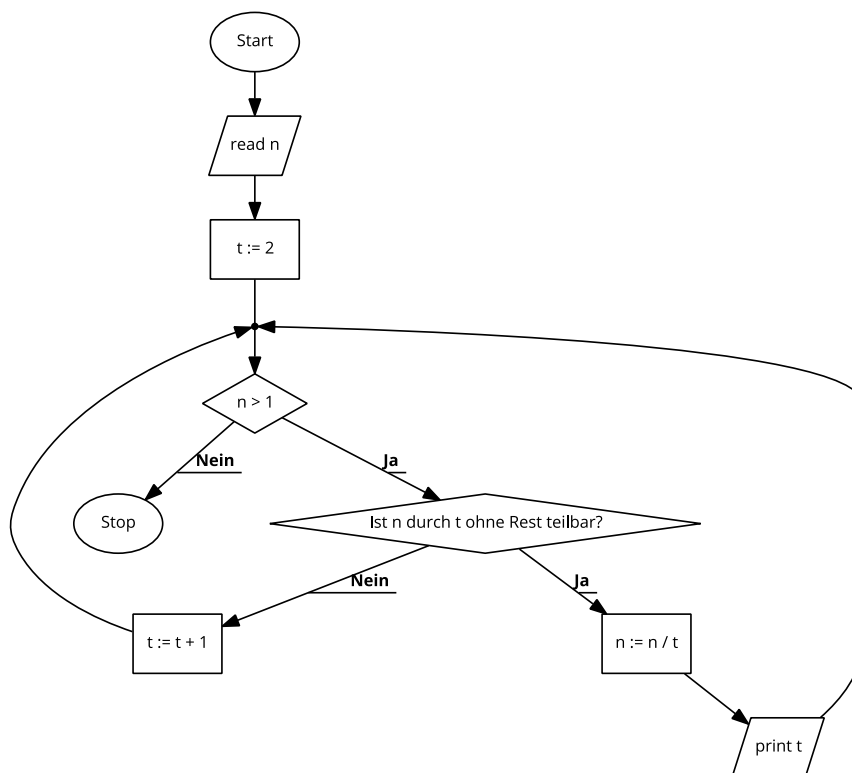


Abbildung 5. Programmablaufplan zur Primfaktorzerlegung

Schreibtischtest

Es wird das Programm mit der Zahl $60 = 2 \cdot 2 \cdot 3 \cdot 5$ getestet.

Schritt	n	t	Ausgabe	n > 1	t teilt n
0	60	2	-	WAHR	WAHR
1	30	2	2	WAHR	WAHR
2	15	2	2	WAHR	FALSCH
3	15	3	-	WAHR	WAHR

Schritt	n	t	Ausgabe	n > 1	t teilt n
4	5	3	3	WAHR	FALSCH
5	5	4	-	WAHR	FALSCH
6	5	5	-	WAHR	WAHR
7	1	5	5	FALSCH	-

4.2. Aufgabe 4: Primzahlen

Eine natürliche Zahl ist eine Primzahl, wenn sie nur durch 1 und durch sich selbst ohne Rest teilbar ist.

Die Primzahlen sind 2, 3, 5, 7, 11, 13, 17, 19, 23, ...

Als einfaches Verfahren, um zu prüfen, ob eine Zahl n eine Primzahl ist, kann man daher bei allen Zahlen zwischen 1 und der Zahl n testen, ob diese die Zahl n ohne Rest teilen. Falls ein Teiler gefunden wurde kann das Verfahren abgebrochen werden, da die Zahl n keine Primzahl ist. Falls kein Teiler gefunden wird, ist die Zahl n eine Primzahl.

Testdaten

Als Testdaten stehen die [Primzahlen von 2 bis 100.000](#) zu Verfügung.

5. Zählschleife

5.1. Beispiel 5: ISBN-10

Bücher werden durch die Internationale Standardbuchnummer (ISBN) eindeutig gekennzeichnet. Bis 2006 wurde dazu eine zehnstellige Nummer verwendet, die eine Prüfziffer enthält. Diese Prüfziffer berechnet man, indem man die erste Ziffer mit eins multipliziert, die zweite mit zwei usw. bis zur neunten Ziffer und die Ergebnisse addiert. Diese Summe wird anschließend durch 11 dividiert. Aus dem Rest dieser Division wird die Prüfziffer gebildet. Falls der Rest 10 ist, wird das Zeichen 'X' verwendet.

Beispiel: ISBN 3-8171-2004-4

Die ersten neun Ziffern sind 3-8171-2004.

$$3 \cdot 1 + 8 \cdot 2 + 1 \cdot 3 + 7 \cdot 4 + 1 \cdot 5 + 2 \cdot 6 + 0 \cdot 7 + 0 \cdot 8 + 4 \cdot 9 = 103$$

$$103 = 9 \cdot 11 \text{ Rest } 4$$

Daher hat die ISBN 3-8171-2004-4 die Prüfziffer 4.

Pseudocode

Die Ziffern der ISBN werden als Feld Z behandelt.

Quellcode 5. Pseudocode zur Berechnung der Prüfziffer für die ISBN-10

```
program ISBN10checkDigit
  summe := 0;
  for i := 1 to 9
    produkt = Z[i] * i;
    summe := summe + produkt;
  p := Rest der Division zwischen summe und 11;
  if p = 10
    print 'X';
  else
    print p;
end program ISBN10checkDigit
```

Programmablaufplan

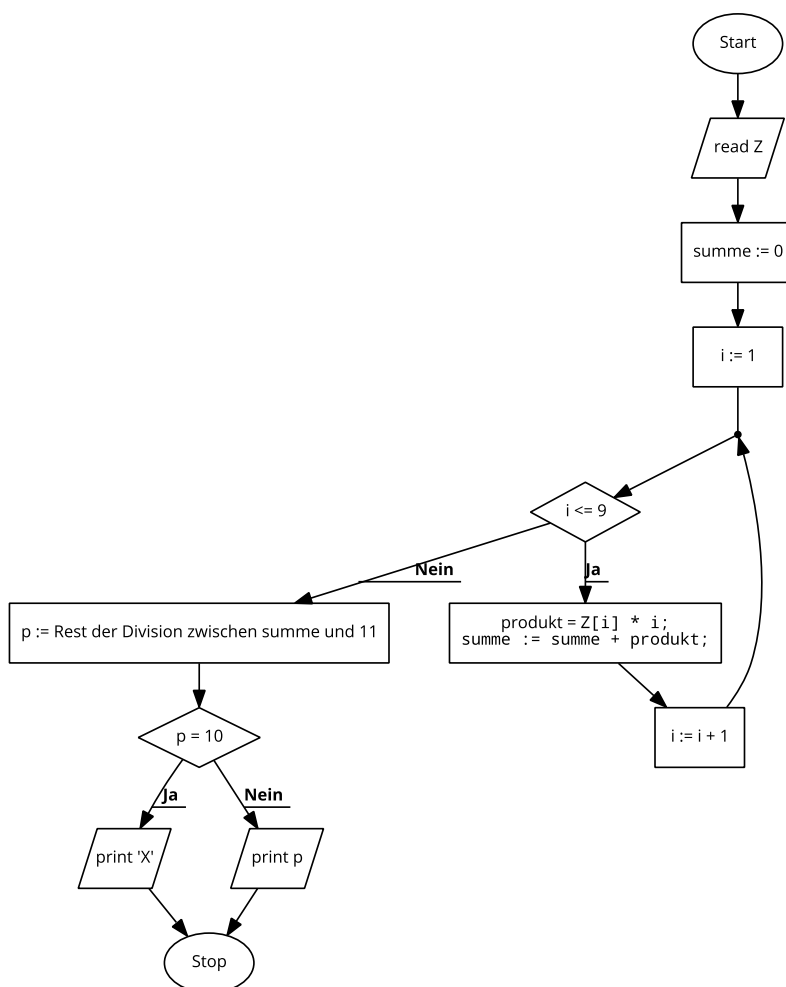


Abbildung 6. Programmablaufplan zur Berechnung der Prüfziffer für die ISBN-10

Schreibtischtest

Es wird das Programm mit der ISBN 3-8273-7046-9 getestet.

Schritt	Z	i	Z[i]	produkt	summe	p	Ausgabe
0	[3,8,2,7,3,7,0,4,6]	-	-	-	0	-	-
1	[3,8,2,7,3,7,0,4,6]	1	3	3	3	-	-
2	[3,8,2,7,3,7,0,4,6]	2	8	16	19	-	-
3	[3,8,2,7,3,7,0,4,6]	3	2	6	25	-	-
4	[3,8,2,7,3,7,0,4,6]	4	7	28	53	-	-
5	[3,8,2,7,3,7,0,4,6]	5	3	15	68	-	-
6	[3,8,2,7,3,7,0,4,6]	6	7	42	110	-	-
7	[3,8,2,7,3,7,0,4,6]	7	0	0	110	-	-
8	[3,8,2,7,3,7,0,4,6]	8	4	32	142	-	-
9	[3,8,2,7,3,7,0,4,6]	9	6	54	196	-	-
10	[3,8,2,7,3,7,0,4,6]	-	-	-	196	9	9

5.2. Aufgabe 5: Pharmazentralnummer

Die Pharmazentralnummer (PZN) ist ein auf in deutschland erhältlichen Arzneimitteln angebrachter Identifikationsschlüssel. Sie besteht derzeit aus acht Ziffern, die letzte Stelle ist eine Prüfziffer. Ähnlich wie bei der ISBN-10 werden die Ziffern von links nach rechts aufsteigend mit den Zahlen 1 bis 7 multipliziert und aufsummiert. Anschließend wird der Rest der Division zwischen der Summe und 11 berechnet. Sollte der Rest 10 ergeben, wird die PZN nicht vergeben.

Beispiel: PZN 10024970

Die ersten sieben Ziffern sind 1002497.

$$1 \cdot 1 + 0 \cdot 2 + 0 \cdot 3 + 2 \cdot 4 + 4 \cdot 5 + 9 \cdot 6 + 7 \cdot 7 = 132$$

$$132 = 12 \cdot 11 \text{ Rest } 0$$

Die PZN 10024970 hat daher die Prüfziffer 0.

Testdaten

Die folgenden Pharmazentralnummern können als Testdaten verwendet werden.

```
07728561  
10201099  
00040554  
00040548  
03806873  
01894063  
10203595  
10203603  
01743631  
03227112  
03464237  
04356248
```