Übung zu Programmkonstrukten: Lösung

Klaus Kusche, 2010 / 2011

Gib den entsprechenden Programmausschnitt an (am Papier / an der Tafel / mündlich).
"Erfinde" passende Variablennamen, wenn du welche brauchst
(die Deklarationen musst du nicht hinschreiben).

1.) Zähle zu **n** 10 dazu.

```
n = n + 10;   oder   n += 10;
```

2.) Wenn die berechnete Zahl negativ ist, merken wir uns -1 in **vorz**, sonst 1.

```
if (z < 0) {
  vorz = -1;
} else {
  vorz = 1;
}
```

oder

```
vorz = (z < 0) ? -1 : 1;
```

3.) Die erste Zahl auf der Befehlszeile ist unsere Länge, die zweite die Breite.

```
len = atof(argv[1]);
bre = atof(argv[2]);
```

(oder **atoi**, wenn **len** und **bre** nicht **double**, sondern **int** sind)

4.) Dividiere **n** immer wieder durch 2, solange es gerade ist.

```
while (n % 2 == 0) {
  n = n / 2;    (oder n /= 2; )
}
```

5.) Wenn Minimum und Maximum gleich sind, soll das Programm einfach enden.

```
if (min == max) {
  return 0;    (oder exit(EXIT_SUCCESS); )
}
```

6.) Gib alle Worte auf der Befehlszeile (ohne Programmname) einzeln
zeilenweise und durchnummeriert aus, und zwar in umgekehrter Reihenfolge.

```
for (i = argc - 1; i > 0; --i) {
  printf("%d.: %s\n", i, argv[i]);
}
```

7.) Zähle alle Zahlen von 1 bis **n** zusammen.

```
sum = 0;
for (i = 1; i <= n; ++i) {
  sum += i;
}
```

8.) Streiche die letzte Stelle von **n** weg.

```
n = n / 10;
```
oder `n /= 10;`

9.) Mach eine Schleife über alle Zahlen in der Befehlszeile und zähle sie zusammen.

```
sum = 0;
for (i = 1; i < argc; ++i) {
  sum += atoi(argv[i]);
}
```

10.) Gib die Länge, die Breite und deren Produkt aus.

```
printf("Länge %f, Breite %f, Fläche %f\n",
       len, bre, len * bre);
```

11.) Ersetze **x** durch seinen Kehrwert.

```
x = 1 / x;
```

12.) Unsere Fläche berechnen wir als Länge mal Breite.

```
flaeche = len * bre;
```

13.) Mach eine Tabelle mit den Quadraten und dritten Potenzen der Zahlen 1 bis 100.

```
for (i = 1; i <= 100; ++i) {
  printf("%6d %6d %6d\n", i, i * i, i * i * i);
}
```

14.) In **z** speichern wir die Summe von **x** und **y**.

```
z = x + y;
```

15.) Drehe das Vorzeichen von **a** und **b** um.

```
a = -a; b = -b;
```

16.) Unser neues **n** ist das alte **n** mal 10 plus die soeben berechnete Ziffer.

```
n = n * 10 + z;
```

17.) Probiere alle Zahlen von 2 aufwärts, bis du eine hast, die **n** glatt teilt.

```
for (i = 2; n % i != 0; ++i) {}
```

18.) Dasselbe, aber probiere nur die ungeraden Zahlen ab 3.

```
for (i = 3; n % i != 0; i += 2) {}
```

19.) Solange **a** größer als **b** ist, ziehe **b** von **a** ab.

```
while (a > b) {
  a -= b;
}
```
(oder `a = a - b;`)

20.) Unser Anfangswert für **i** ist 100.

```
i = 100;
```

21.) Setze **m** auf den Mittelwert von **a** und **1/a**.

```
m = (a + 1 / a) / 2;
```

22.) Gib **1/x** aus, wenn **x** von 0 verschieden ist. Sonst gib 0 aus.

```
if (x != 0) {
  printf("%f\n", 1 / x);
} else {
  printf("0\n");
}
```

oder

```
printf("%f\n", (x != 0) ? (1 / x) : 0);
```

23.) Wenn **n** einstellig ist, multipliziere es mit 10.

```
if (n < 10) {
  n *= 10;
}
```

24.) Gib die (ganzen) Zahlen von **j** bis **k** (jeweils einschließlich) aus.

```
for (i = j; i <= k; ++i) {
  printf("%d\n", i);
}
```

25.) In **a** speichern wir unser **x** ohne Vorzeichen.

```
a = abs(x);
```
(oder **fabs**, wenn wir es mit **double**-Zahlen zu tun haben)

26.) Solange die Länge zumindest doppelt so groß wie die Breite ist:
Halbiere die Länge und verdopple die Breite.

```
while (len >= 2 * bre) {
  len /= 2;
```
(oder `len = len / 2;`)
```
  bre *= 2;
```
(oder `bre = bre * 2;`)
```
}
```

27.) Merke dir die letzte Stelle von **n** als **s**.

```
s = n % 10;
```

28.) Ersetze den kleineren unserer beiden Zwischenwerte durch 1.

```
if (z1 < z2) {
  z1 = 1;
} else {
  z2 = 1;
}
```

29.) Gib **n** Sternchen aus.

```
for (i = 1; i <= n; ++i) {
  printf("*");
}
```

oder

```
for (i = 0; i < n; ++i) {
  printf("*");
}
```

oder

```
for ( ; n > 0; --n) {
  printf("*");
}
```

30.) Erhöhe **k** um 1.

```
++k;
```

31.) Berechne **q** als $a^2 + b^2$.

```
q = a * a + b * b;
```

32.) Wenn **a/b** vom Absolutbetrag her kleiner 1 ist, gib "Zu klein" aus.

```
if (fabs(a / b) < 1) {
  printf("Zu klein!\n");
}
```

oder

```
if ((a / b > -1) && (a / b < 1)) ...
```

33.) Teile **b** wiederholt durch **n**, bis es kleiner als **n** ist.

```
while (b >= n) {
  b /= n;
}
```

34.) Unser neues **r** ist der Rest vom alten **r** dividiert durch **i**.

```
r = r % i;
```

35.) Speichere den Mittelwert von bisherigem Zwischenergebnis und aktueller Eingabe als neues Zwischenergebnis.

```
zwWert = (zwWert + eing) / 2;
```

36.) Als Limit merken wir uns die größere der beiden Zahlen **a** und **b**.

```
if (a > b) {
  limit = a;
} else {
  limit = b;
}
```

oder

```
limit = (a > b) ? a : b;
```

37.) Solange **x** kleiner als **y** ist, quadrieren wir **x** immer wieder.

```
while (x < y) {
  x = x * x;
}
```

Und jetzt länger / trickreicher / schwerer:
(bei den meisten dieser Beispiele gibt es viele verschiedene Lösungen!)

38.) Gehe die Zahlen von **n - 1** abwärts durch und gib alle die aus, die **n** teilen.

```
for (i = n - 1; i > 0; --i) {
  if (n % i == 0) {
    printf("%d ", i);
  }
}
```

39.) Ermittle, wie viele Stellen (Ziffern) die Zahl **n** hat.

```
for (stellen = 0; n > 0; ++stellen) {
  n /= 10;
}
```
(wenn 0 als Zahl mit 0 Stellen gilt)

oder

```
for (stellen = 1; n > 9; ++stellen) {
  n /= 10;
}
```
(wenn 0 als Zahl mit 1 Stelle gilt)

40.) Berechne die Summe aller Zweierpotenzen die kleiner als **x** sind.

```
sum = 0;
for (i = 1; i < x; i *= 2) {
  sum += i;
}
```

41.) Erhöhe **x** in Zehnerschritten, bis $\mathbf{x}^2$ größer als **y** ist.

```
while (x * x <= y) {
  x += 10;
}
```

42.) Berechne das Produkt
a/b * (a+1)/(b+1) * (a+2)/(b+2) * ... * (a+n)/(b+n).

```
prod = 1;
for (i = 0; i <= n; ++i) {
  prod *= (a + i) / (b + i);
}
```

43.) Wenn **b** größer als **a** ist, vertausche **a** und **b** (es gibt in C nichts zum direkten Vertauschen, verwende eine zusätzliche Variable und weise “im Kreis herum” zu!).

```
if (b > a) {
  tmp = a;
  a = b;
  b = tmp;
}
```

44.) Ermittle, wie oft man **n** verdoppeln muss, bis es größer als **endwert** ist.

```
for (cnt = 0; n <= endwert; ++cnt) {
  n *= 2;
}
```

45.) Wiederhole Folgendes, bis **n** gleich 1 ist, und zähle die Anzahl der Schritte mit:
Wenn **n** gerade ist, teile es durch 2, sonst multipliziere es mit 3 und zähle 1 dazu[1].

```
for (cnt = 0; n > 1; ++cnt) {
  if (n % 2 == 0) {
    n = n / 2;
  } else {
    n = n * 3 + 1;
  }
}
```

1 Der deutsche Mathematiker Collatz hat vor dem 2. Weltkrieg vermutet,
dass diese Schleife für jeden Startwert von **n** irgendwann einmal endet.
Das ist bis heute unbewiesen und eines der bekanntesten offenen Probleme der Mathematik.