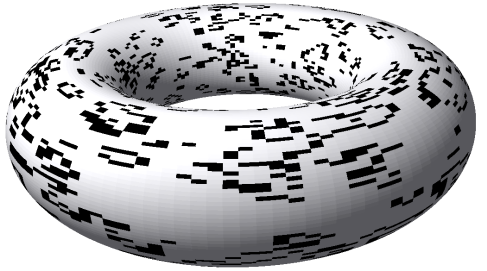
Einführung in die Programmierung



Tutorium 5

Wintersemester 2025/2026

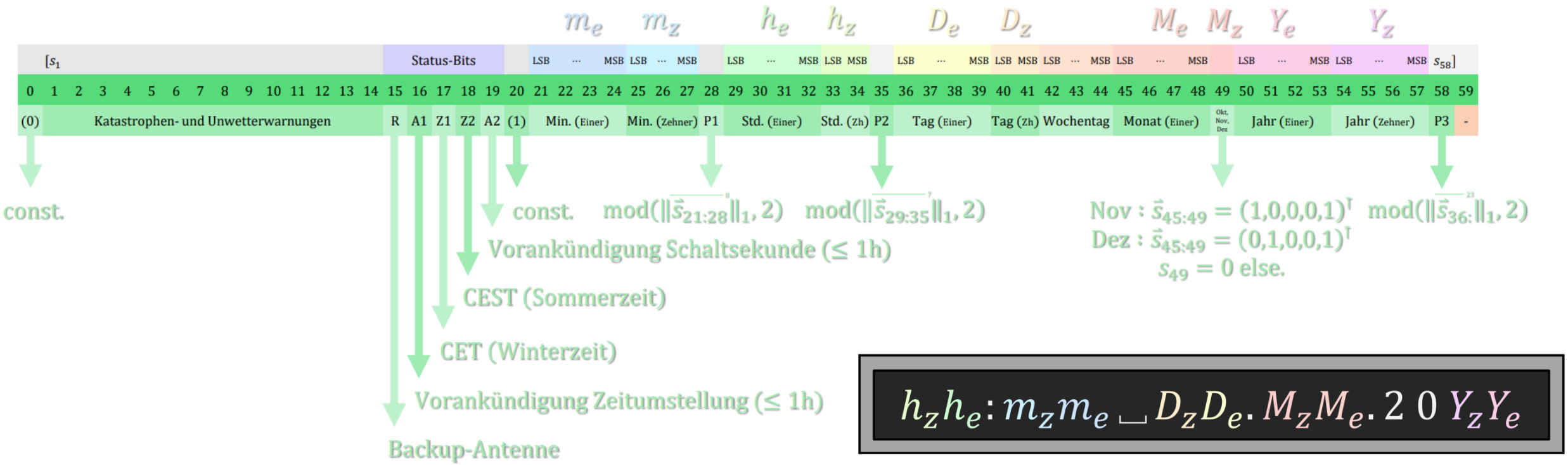
Arbeitsgruppe Systemsoftware
Angewandte Informatik 12

technische universität
dortmund

Arbeitsgruppe
Systemsoftware

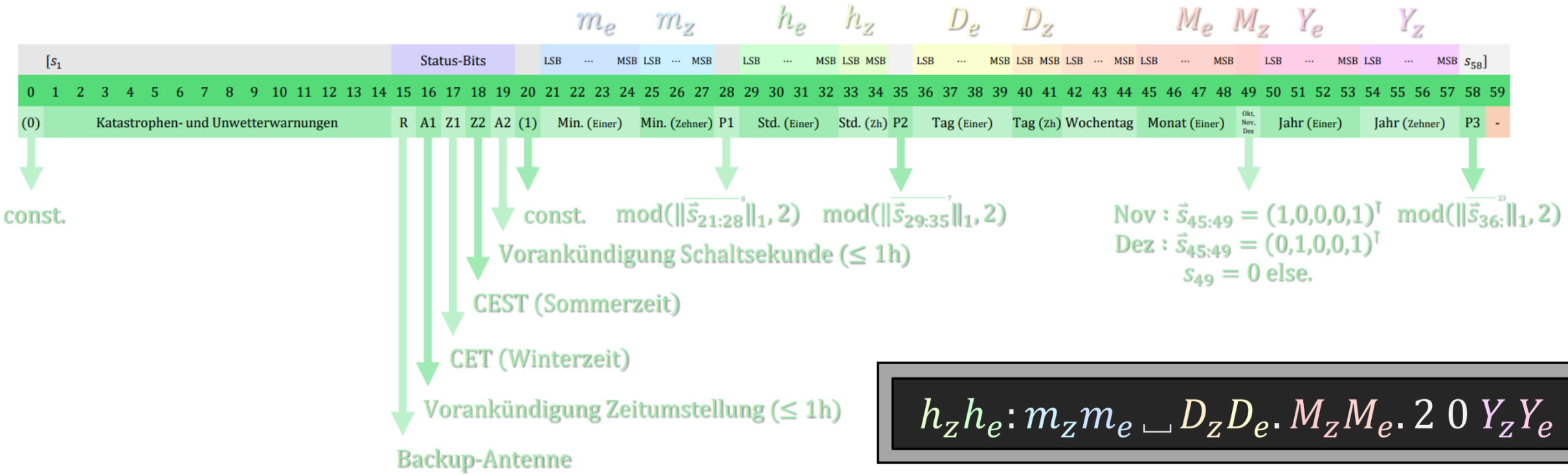
DCF77-Sequenz – Aufgabenstellung

Implementieren Sie einen Decoder für das DCF77-Funksignal, der aus einer 59-Bit-Sequenz Datum und Uhrzeit extrahiert (BCD-Decodierung) und daraus den korrekten UNIX-Zeitstempel berechnet.



$$h_z h_e : m_z m_e \; \_ \; D_z D_e . M_z M_e . 2\,0\; Y_z Y_e$$

59-Bit-lange Funkübertragung für aktuelle Zeit in binär codierten Dezimalzahlen (BCD)

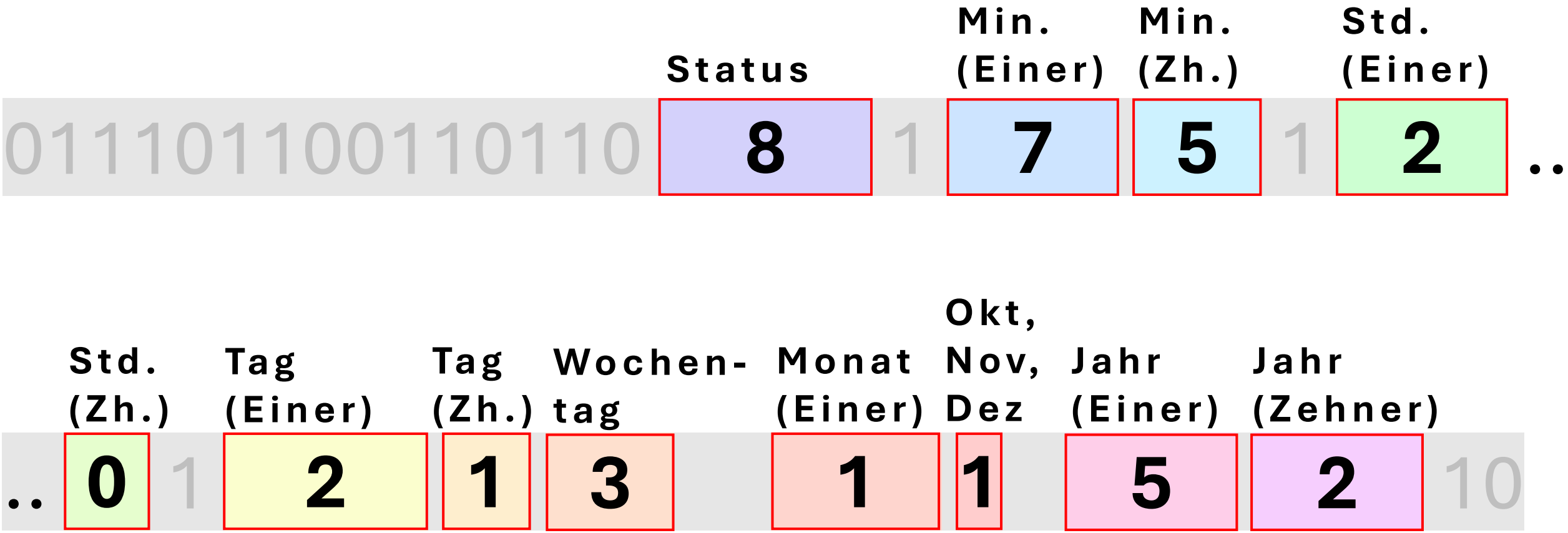


DCF77-Sequenz – Idee

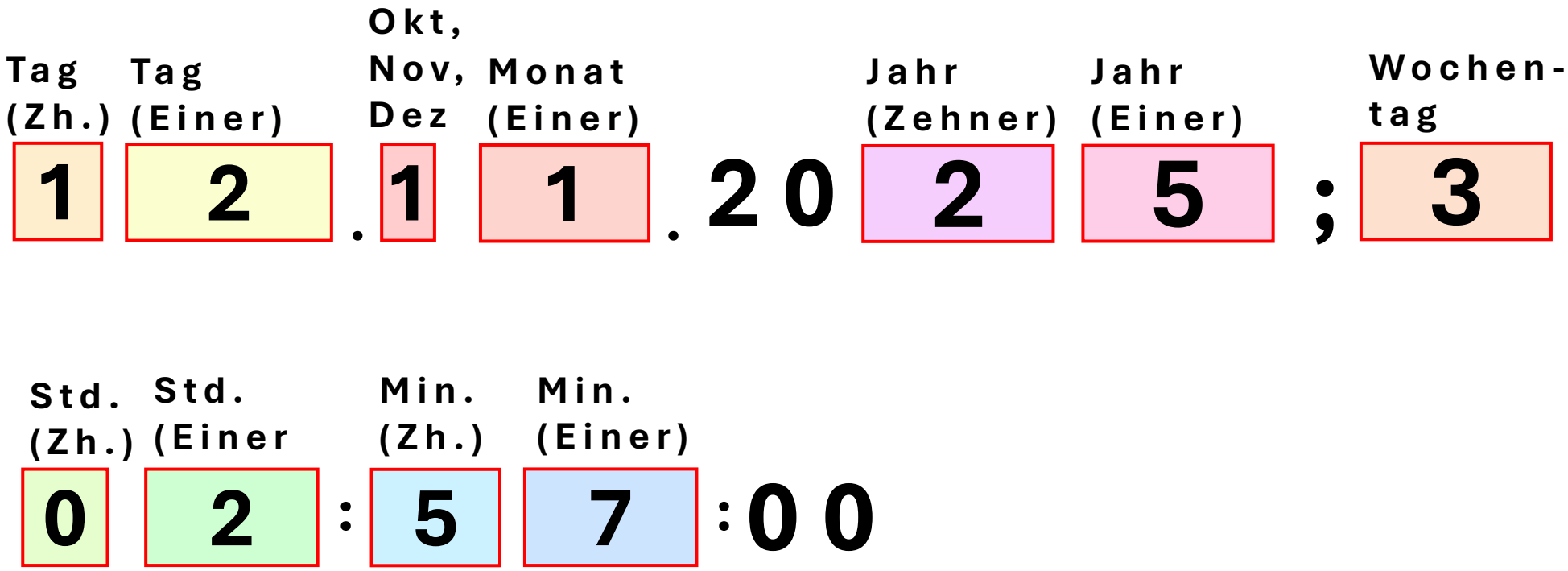
	Status		Min. (Einer)	Min. (Zh.)		Std. (Einer)	
011101100110110	**00010**	1	**1110**	**101**	1	**0100**	..

	Std. (Zh.)		Tag (Einer)	Tag (Zh.)	Wochen-tag	Monat (Einer)	Okt, Nov, Dez	Jahr (Einer)	Jahr (Zehner)	
..	**00**	1	**0100**	**10**	**110**	**1000**	**1**	**1010**	**0100**	10

DCF77-Sequenz – Idee



DCF77-Sequenz – Idee



DCF77-Sequenz – Code

Aufgabe 1

CODEAUSSCHNITT | TODO1

```
7   unsigned int bcd(unsigned long long int x, int from, int to) {
8       unsigned int val = 0, mult = 1;
9       int step = (to >= from) ? 1 : -1;
10
11       //!TODO1
12       for (int i = from; i != to + step; i += step) {
13           if ((x >> i) & 1ULL) val += mult;
14               mult <<= 1;
15       }
16   return val;
17   }
```

DCF77-Sequenz – Code

Aufgabe 1

CODEAUSSCHNITT | TODO2

```
34    unsigned int min_einer = bcd(dfc, 21, 24);
35    unsigned int min_zehner = bcd(dfc, 25, 27);
36    unsigned int minute = min_zehner * 10 + min_einer;
37
38    unsigned int hr_einer = bcd(dfc, 29, 32);
39    unsigned int hr_zehner = bcd(dfc, 33, 34);
40    unsigned int hour = hr_zehner * 10 + hr_einer;
41
42    unsigned int day_einer = bcd(dfc, 36, 39);
43    unsigned int day_zehner = bcd(dfc, 40, 41);
44    unsigned int day = day_zehner * 10 + day_einer;
```

...

DCF77-Sequenz – Code

Aufgabe 1

CODEAUSSCHNITT | TODO2 (FORTSETZUNG)

```
46    unsigned int weekday = bcd(dfc, 42, 44);
47
48    unsigned int month_einer = bcd(dfc, 45, 48);
49    unsigned int month_zehner = bcd(dfc, 49, 49);
50    unsigned int month = month_zehner * 10 + month_einer;
51
52    unsigned int yr_einer = bcd(dfc, 50, 53);
53    unsigned int yr_zehner = bcd(dfc, 54, 57);
54    unsigned int year = 2000 + yr_zehner * 10 + yr_einer;
```

DCF77-Sequenz – Code

CODEAUSSCHNITT | TODO3

```
57    unsigned long long int t = 0;
58    for (int y = 1970; y < year; y++) {
59        t += (unsigned long long int)(is_leap(y) ? 366ULL : 365ULL) * 86400ULL;
60    }
61
62    if (is_leap(year)) mdays[1] = 29;
63    for (int m = 0; m < month - 1; m++) {
64        t += (unsigned long long int)mdays[m] * 86400ULL;
65    }
```

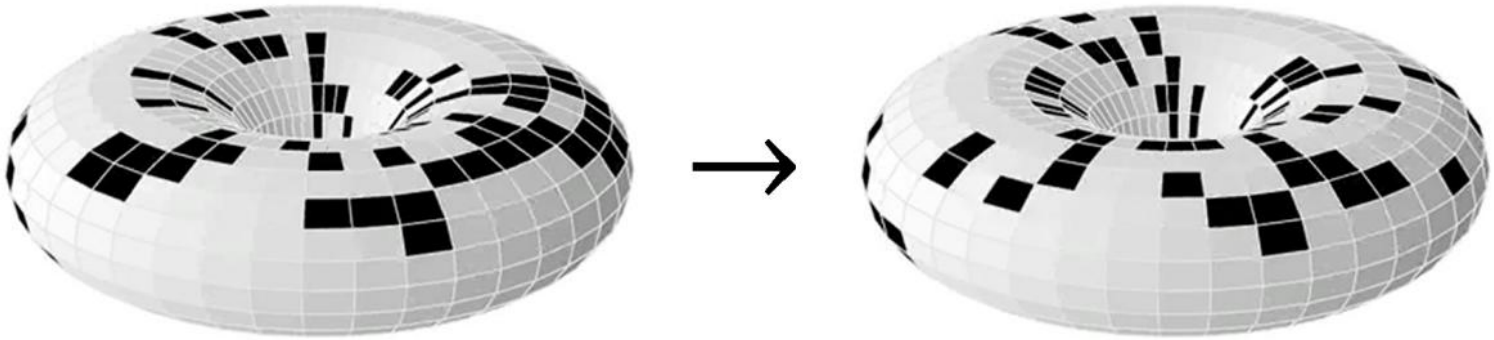
...

DCF77-Sequenz – Code

CODEAUSSCHNITT | TODO3 (FORTSETZUNG)

```
67    t += (unsigned long long int)(day - 1) * 86400ULL;
68    t += (unsigned long long int)hour * 3600ULL;
69    t += (unsigned long long int)minute * 60ULL;
70    t -= 60ULL*60ULL;
```

Implementiere Sie die Nachbarzählung mit periodischen Rändern (Torus) und die Generationen-Berechnung gemäß Conway-Regeln, um die Simulation des Game of Life korrekt durchzuführen.



Game of Life – Code (Beispiel)

Aufgabe 2

CODEAUSSCHNITT | TODO1

```
38   int count_neighbors(int x, int y) {
39       int count = 0;
40
41       //!TODO1
40       for (int dx = -1; dx <= 1; dx++) {
41           for (int dy = -1; dy <= 1; dy++) {
42               if (dx == 0 && dy == 0) continue;
43               int nx = (x + dx + GRID_SIZE) % GRID_SIZE;
44               int ny = (y + dy + GRID_SIZE) % GRID_SIZE;
45               if (grid[nx][ny]) count++;
46           }
47       }
48       return count;
49   }
```

Game of Life – Code

Aufgabe 2

CODEAUSSCHNITT | TODO2

```
53   void update_grid(double currentTime) {
54      bool new_grid[GRID_SIZE][GRID_SIZE] = {0};
55      for (int x = 0; x < GRID_SIZE; x++) {
56         for (int y = 0; y < GRID_SIZE; y++) {
57            int neighbors = count_neighbors(x, y);
58            if (grid[x][y]) {
59               new_grid[x][y] = (neighbors == 2 || neighbors == 3);
60            } else {
61               new_grid[x][y] = (neighbors == 3);
62            }
63         }
64      }
65   memcpy(grid, new_grid, sizeof(grid));
66   lastUpdateTime = currentTime;
67   }
```