Thema, Ziele: CRC Berechnung und C++ Implementation

Bemerkung:

In dieser Lösung werden die Bytes MSB-First codiert.

Aufgabe 1: CRC-8 Berechnungsbeispiel

Papierübgung

Berechnen Sie die Checksumme für den unten abgebildeten Bytestream. Verwenden Sie dafür das Generatorpolynom $G = x^8 + x^2 + x^1 + 1$ (entspricht CRC-8).

```
Message:       0 0 1 0 1 1 0 1 = 0x2D
Polynomial:  1 0 0 0 0 0 1 1 1 = 0x107
CRC:           1 1 0 0 0 0 1 1 = 0xC3
```

Berechnung:

```
        1 0 1 1 0 1 0 0 0           = CRC
        1 0 0 0 0 0 1 1 1           = XOR Polynomial
        ---------------------------
            1 1 0 1 1 1 1 0 0       = CRC
            1 0 0 0 0 0 1 1 1       = XOR Polynomial
            -----------------------
              1 0 1 1 1 0 1 1 0     = CRC
              1 0 0 0 0 0 1 1 1     = XOR Polynomial
              ---------------------
                  1 1 1 0 0 0 1 0 0 = CRC
                  1 0 0 0 0 0 1 1 1 = XOR Polynomial
                  -----------------
                    1 1 0 0 0 0 1 1 = CRC = 0xC3
                    ===============
```

Aufgabe 2: Komplexitätsbetrachtungen

Das verwendete Generatorpolynom ist dasselbe wie in Aufgabe 1.

```
Message 1: 1 1 0 1 1 0 0 1  0 0 0 0 0 0 0 1 wurde korrekt empfangen
    1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 1
    ================================
    1 1 0 1 1 0 0 1 0               = CRC
    1 0 0 0 0 0 1 1 1               = XOR Polynomial
    -------------------------------
      1 0 1 1 0 1 0 1 0             = CRC
      1 0 0 0 0 0 1 1 1             = XOR Polynomial
      -----------------------------
          1 1 0 1 1 0 1 0 0         = CRC
          1 0 0 0 0 0 1 1 1         = XOR Polynomial
          -------------------------
            1 0 1 1 0 0 1 1 0       = CRC
            1 0 0 0 0 0 1 1 1       = XOR Polynomial
            -----------------------
                1 1 0 0 0 0 1 0 0   = CRC
                1 0 0 0 0 0 1 1 1   = XOR Polynomial
                -------------------
                  1 0 0 0 0 0 1 1 1 = CRC
                  1 0 0 0 0 0 1 1 1 = XOR Polynomial
                  -----------------
                    0 0 0 0 0 0 0 0 = no residual, message 1 valid
```

```
Message 2: 0 1 0 1 0 0 0 1  0 1 1 1 0 1 1 1 wurde falsch empfangen


    0 1 0 1 0 0 0 1 0 1 1 1 0 1 1 1
  ===================================
      1 0 1 0 0 0 1 0 1              = CRC
      1 0 0 0 0 0 1 1 1              = XOR Polynomial
      -----------------------------
          1 0 0 0 0 1 0 1 1          = CRC
          1 0 0 0 0 0 1 1 1          = XOR Polynomial
          -------------------------
                    1 1 0 0 0 0 0 0  = residual 0xC0, message 2 corrupt

Message 3: 0 1 1 0 0 0 0 1  0 0 1 0 0 0 0 0 wurde korrekt empfangen


    0 1 1 0 0 0 0 1 0 0 1 0 0 0 0 0
  ===================================
      1 1 0 0 0 0 1 0 0              = CRC
      1 0 0 0 0 0 1 1 1              = XOR Polynomial
      -----------------------------
        1 0 0 0 0 0 1 1 1            = CRC
        1 0 0 0 0 0 1 1 1            = XOR Polynomial
        -------------------------
                    0 0 0 0 0 0 0 0  = no residual, message 3 valid
```

Die zweite Message wurde falsch empfangen da bei der CRC Berechnung ein Rest von `0xC0` übrig bleibt. Entweder ist das Daten- oder das Prüfsummenbyte korrupt. Der Empfänger muss in einem solchen Fall ein NACK zurückgeben, um die Message noch einmal anzufordern.

Aufgabe 3: C++ Implementation

siehe ./Loesung/A3

Wird das Programm ohne Argument aufgerufen, berechnet es die Prüfsumme von {0xd9, 0x51, 0x61}. Mit einem gültigen Dateinamen als Argument wird diese Datei in einen Byte-Buffer gelesen und die Prüfsumme über den gesamten Buffer berechnet.

Aufgabe 4: C++ Implementation mit einer Look-up Tabelle

siehe ./Loesung/A4

Wird das Programm ohne Argument aufgerufen, berechnet es die Prüfsumme von {0xd9, 0x51, 0x61}:

```
$ ./crcTest.exe
CRC-8 CCITT Bit um Bit, einfach aber ineffizient

Datei:             datafile.txt
Anzahl Datenbytes: 72125
CRC-8 CCITT:       0x10
CRC-8 CCITT Bit um Bit, einfach aber ineffizient

CRC-8 CCITT der einzelnen Datenbytes:
Byte 00: crc(0xd9) = 0x01

Byte 01: crc(0x51) = 0xb0

Byte 02: crc(0x61) = 0x20
```

```
CRC-8 CCITT von allen Datenbytes:
crc(0xd95161) = 0x2c
```

Mit einem gültigen Dateinamen als Argument wird diese Datei in einen Byte-Buffer gelesen und die Prüfsumme über den gesamten Buffer berechnet.

```
$ ./crcTest.exe datafile.txt
CRC-8 CCITT Bit um Bit, einfach aber ineffizient

Datei:             datafile.txt
Anzahl Datenbytes: 72125
CRC-8 CCITT:       0x10
```