Codierungstheorie

Codes im Alltag

- Morsecode
- Barcode
- Binärcode
- QR-Code

… / --- / …

Beispiel – Fehler bei der Übertragung

- f: B → B^3
  - Drei Ziffern
  - Fehler erkennen
  - Fehler ausbessern

0 → **Codierer** → 000

000 → **Übertragung** → 011 2 Fehler

000 → **Übertragung** → 001 1 Fehler

001 → **Decodierer** → 0

Codieren / Decodieren

codieren.nb *

Wolfram Mathematica | PRODUCT TRIAL

```
Codieren[a_, u_, p_] := Module [{f, L, m, n},
  m = Length[a];
  n = Length[u];
  f = Sum[a[[i + 1]] x^i, {i, 0, m - 1}];
  L = Table[Mod[f /. x → u[[i]], p], {i, 1, n}];
  Return[L]
 ]

c = Codieren[{0, 1, 3, 1334, 4}, {1, 2, 3, 4, 5, 6}, 31]
```

```
Codieren[{0, 1, 3, 1334, 4}, {1, 2, 3, 4, 5, 6}, 31]
```

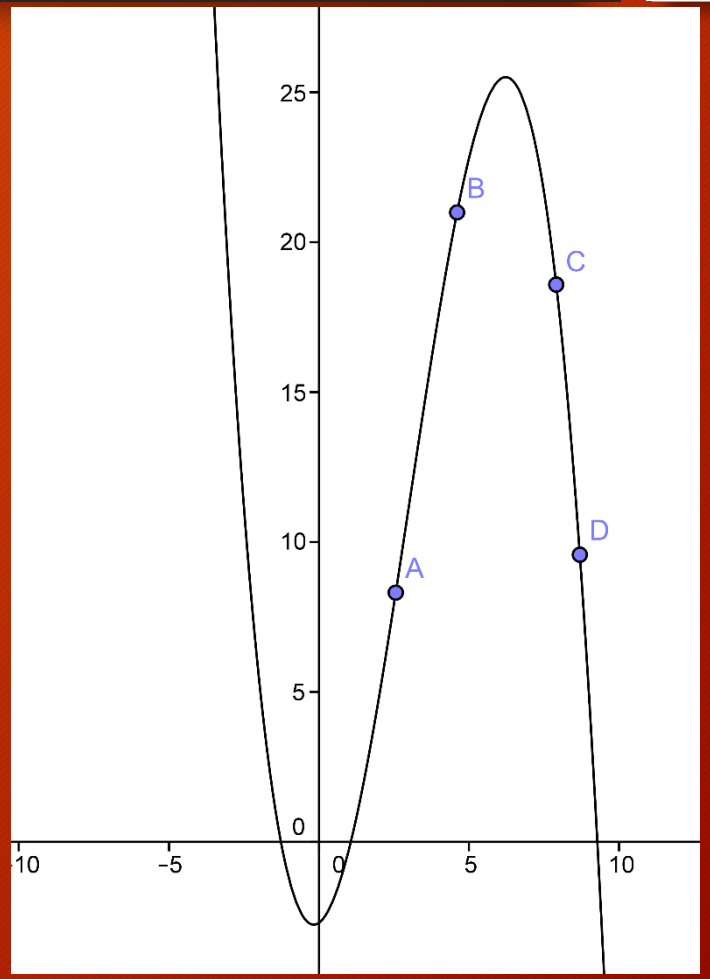
```
Decodieren[c_, u_, m_, p_] := Module[{f},
  f = InterpolatingPolynomial[Inner[List, u, c, List][[1 ;; m]], x, Modulus → p];
  Return[Mod[CoefficientList[f, x], p]]
 ]

Decodieren[{1342, 10750, 36372, 86452, 169330, 293442}, {1, 2, 3, 4, 5, 6}, 5, 31]
```

```
{0, 1, 3, 1, 4}
```

```
{0, 1, 3, 1, 4}
```

```
a := {3, 9, 13, 20, 20};
u := {1, 2, 4, 6, 7, 8, 10};
m := Length[a];
p := 31;
c := Codieren[a, u, p]
Decodieren[c, u, m, p]
```

```
{3, 9, 13, 20, 20}
```



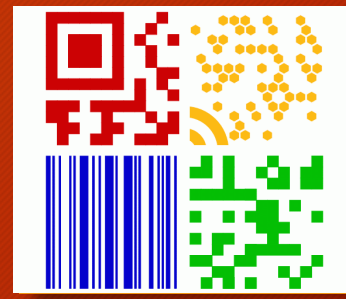
Abschlussaufgabe

Text als Grafik codieren:

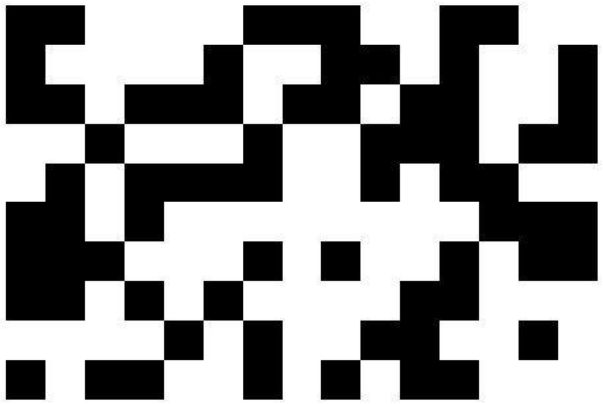
1. Text E
2. → Zahlen 5
3. → Codieren (!)
4. → Binärcode 101
5. → Grafik 

```
Schritt1[T_String] := Module[{T1},
  T1 = ToCharacterCode[T] - 64;
  Return[T1]
 ]

Schritt2[T1_, u_, p_] := Module[{L, f, m, n},
  m = Length[T1];
  n = Length[u];
f = Sum[T1[[i + 1]] x^i, {i, 0, m - 1}];
L = Table[Mod[f /. x → u[[i]], p], {i, 1, n}];
Return[L];
]

Schritt3[L_] := Module[{},
  IntegerDigits[L, 2, 5]
 ]


Gesamt[T_String] := Module[{u, p, s1, s2, s3, s4, s5, s6},
  u = Range[30];
  p = 29;
  s1 = Schritt1[T];
  s2 = Schritt2[s1, u, p];
  s3 = Schritt3[s2];
  s4 = Partition[s3, 3];
  s5 = Flatten[#] & /@ s4;
  s6 = MatrixPlot[s5, ColorFunction → "Monochrome", Frame → False];
  Return[s6]
 ]
```

Abschlussaufgabe



```
Gesamt["JULIA"]
```



```
Umwandlung[Dateiname_, blocklength_, width_, dropu_, dropl_, dropr_, dropd_] := Module[{B, B1, B2, B3, B4, CM, CM1, CM2, c, a, j},
  B = Import[Dateiname, "Data"];
  B1 = B /. {a_, a_, a_} → a;
  B2 = Flatten[#] & /@ B1;
  B3 = B2 /. a_ /; (a < 255 / 2) → 0;
  B4 = B3 /. j_ /; (j > 255 / 2) → 255;
  CM = GetColorMatrix[Flatten[B4], blocklength, width, dropu, dropl, dropr, dropd];
  CM1 = Flatten[CM /. {0 → 1, 255 → 0}];
  CM2 = Partition[CM1, 5];
  c = FromDigits[#, 2] & /@ CM2;
  Return[c]
 ]
```

Abschlussaufgabe

Grafik in Text decodieren

```
Loesen[y_] := Module[{u, m, p, n, k, h, g, L, M, f},
  u = Range[30];
  m = 20;
  p = 29;
  n = Length[y];
  k = Ceiling[(n - m)/2];
  h = Sum[b[i] x^i, {i, 0, k}];
  g = Sum[d[i] x^i, {i, 0, k + m - 1}];
  L = Solve[Table[y[[i]] * (h /. x → u[[i]]) + (g /. x → u[[i]]) == 0, {i, 1, n}], Modulus → p];
  M = Table[C[i] → 1, {i, 1, 100}];
  f = PolynomialQuotient[-g /. L[[1]] /. M, h /. L[[1]] /. M, x, Modulus → p];
  Return[FromCharacterCode[CoefficientList[f, x] + 64]]
 ]

Loesen[c]

JULIA
```