Hamming-Code

Prof. Dr. Dörte Haftendorn, MuPAD 4, Aug 08 Update 27. Aug 08
http://haftendorn.uni-lueneburg.de www.mathematik-verstehen.de
www.mathematik-sehen-und-verstehen.de

++

R.W. Hamming, 1948, Bell Labratories, erster effiziente Fehler-korrigiernde Code
(aus van Lint in Aigner/Behrends: Alles Mathematik, S. 11 ff)

Herleitung der nötigen Funktionen
zusammengefasste Befehle zum Ausprobieren
Spielwiese für eigene Listen

##

```
dreheliste:=proc(g_liste)
            local li,la, i;
            begin
               li:=[];la:=nops(g_liste);
               for i from 1 to nops(g_liste) do
                  li:=li.[g_liste[la+1-i]]
               end_for;
               return(li);
             end_proc:
```

```
sz:=proc(z)
      local li;
      begin
        li:=dreheliste(numlib::g_adic(z, 2)):
        if  z<2 then li:=[0,0,0].li;break;
          elif z<4 then li:=[0,0].li;break;
            elif z<8 then li:=[0].li;break;
        end_if;
       return(li);
     end_proc:
```

```
sz(k) $ k=255..258
```

$$[1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 0, 0, 0, 0, 0, 0, 0], [1, 0, 0, 0, 0, 0, 0, 0, 1], [1, 0, 0, 0, 0, 0,$$

```
dreheliste(numlib::g_adic(11, 2))
```

$$[1, 0, 1, 1]$$

```
dreheliste([1,2,3])
```

$$[3, 2, 1]$$

Die Funktion sz(Dezimalzahl) gibt die zugehörige Dualzahl als Liste zurück.

```
sc:=proc(sz)
      local i,li;
      begin

        li:=sz.[(sz[1]+sz[2]+sz[4]) mod 2];
        li:=li.[(sz[1]+sz[3]+sz[4]) mod 2];
        li:=li.[(sz[2]+sz[3]+sz[4]) mod 2];
        return(li);
      end_proc:
```


Im Folgenden werden ganze Dezimalzahlen unter 16 Hamming-codiert

Im Folgenden werden ganze Dezimalzahlen unter 16 Hamming-codiert
Die Funktion sc(Dualliste) hängt die drei Prüfbits an.

```
c:=sc(sz(11))
```

$[1, 0, 1, 1, 0, 1, 0]$

Die nachfolgende Funktion baut an der Stelle p einen Fehler ein.

```
baueFehler:=proc(c,p)
              local i;
              begin
                if c[p]=0 then c[p]:=1;
                else c[p]:=0;
                end_if;
              return(c);
              end_proc:
```

```
cf:=baueFehler(c,5)
```

$[1, 0, 1, 1, 1, 1, 0]$

Die folgende Prozedur sucht den Fehler und korrigiert ihn.

```
such:=proc(cf)
        local i,f,t,p,c;
        begin
          print("es kam an: ",cf);
          f:=0;p:=0;t:=0;c:=cf;
          if (c[1]+c[2]+c[4]) mod 2 <>c[5] then f:=f+1;p:=5;t:=p;
end_if;
          if (c[1]+c[3]+c[4]) mod 2 <>c[6] then
f:=f+1;p:=6;t:=p+t;end_if;
          if (c[2]+c[3]+c[4]) mod 2 <>c[7] then
f:=f+1;p:=7;t:=p+t;end_if;
          case f
  of 0 do
    print("ok ");break;
  of 1 do
    print("Fehler Platz ",p);c:=baueFehler(c,p);break;
  of 2 do
    case t
      of 11 do  print("Fehler Platz ",1);c:=baueFehler(c,1);break;
      of 12 do  print("Fehler Platz ",2);c:=baueFehler(c,2);break;
      of 13 do  print("Fehler Platz ",3);c:=baueFehler(c,3);break;
     end_case;break;
  of 3 do print("Fehler Platz ",4);c:=baueFehler(c,4);break;
  end_case;
  print("gesendet wird nun: ");
  return(c);
 end_proc:
```

```
cn:=such(cf)
"es kam an: ", [1, 0, 1, 1, 1, 1, 0]
"Fehler Platz ", 5
"gesendet wird nun: "
```

$[1, 0, 1, 1, 0, 1, 0]$

Rückverwandlung

```
c2dez:=proc(cn)
```

```
c2dez:=proc(cn)
              begin
                  dez:=0;
                  for i from 1 to 4 do
                     dez:=dez+c[i]*2^(4-i)
                   end_for;
                  return(dez)
              end_proc:
c2dez(cn)
```

11

##

zusammengefasste Befehle zum Ausprobieren 0 <= z <=15

```
z:=8;
sz(z);
c:=sc(sz(z));
cf:=baueFehler(c,5);
cn:=such(cf);
c2dez(cn);
```

8

[1, 0, 0, 0]

[1, 0, 0, 0, 1, 1, 0]

[1, 0, 0, 0, 0, 1, 0]

```
"es kam an: ", [1, 0, 0, 0, 0, 1, 0]
"Fehler Platz ", 5
"gesendet wird nun: "
```

[1, 0, 0, 0, 1, 1, 0]

8

##

Spielwiese für Listen von Hand

```
such([1,1,1,1,1,1,0])
"es kam an: ", [1, 1, 1, 1, 1, 1, 0]
"Fehler Platz ", 7
"gesendet wird nun: "
```

[1, 1, 1, 1, 1, 1, 1]