Euklid. Algorithmus und multiplikatives Inverses in $\mathbb{Z}_p$

Für jedes $n \in \mathbb{Z}_p$ mit $0 \in n < p$ ist

$$ggT(n, p) = 1$$

wenn p eine Primzahl ist.

Der euklid. Algorithmus und also den ggT 1 liefert:

Bsp. 1

$$17 = 3 \cdot 5 + 2$$
$$5 = 2 \cdot 2 + (1)$$
$$2 = 2 \cdot (1) + 0 \quad \text{stop}$$

ggT, obwohl und links an 0

also:

$$1 = 5 - 2 \cdot 2$$
$$= 5 - 2(17 - 3 \cdot 5) =$$
$$= 5 + 6 \cdot 5 - 2 \cdot 17 =$$
$$= 7 \cdot 5 - 2 \cdot 17 \overset{!}{\equiv} 7 \cdot 5 \mod 17 \; !$$

also ist $7 = 5^{-1} \mod 17$!!

Bsp. 2 39^{-1} in $\mathbb{Z}_{53}$?? 53 ist prim !

isPrime (53) ↵

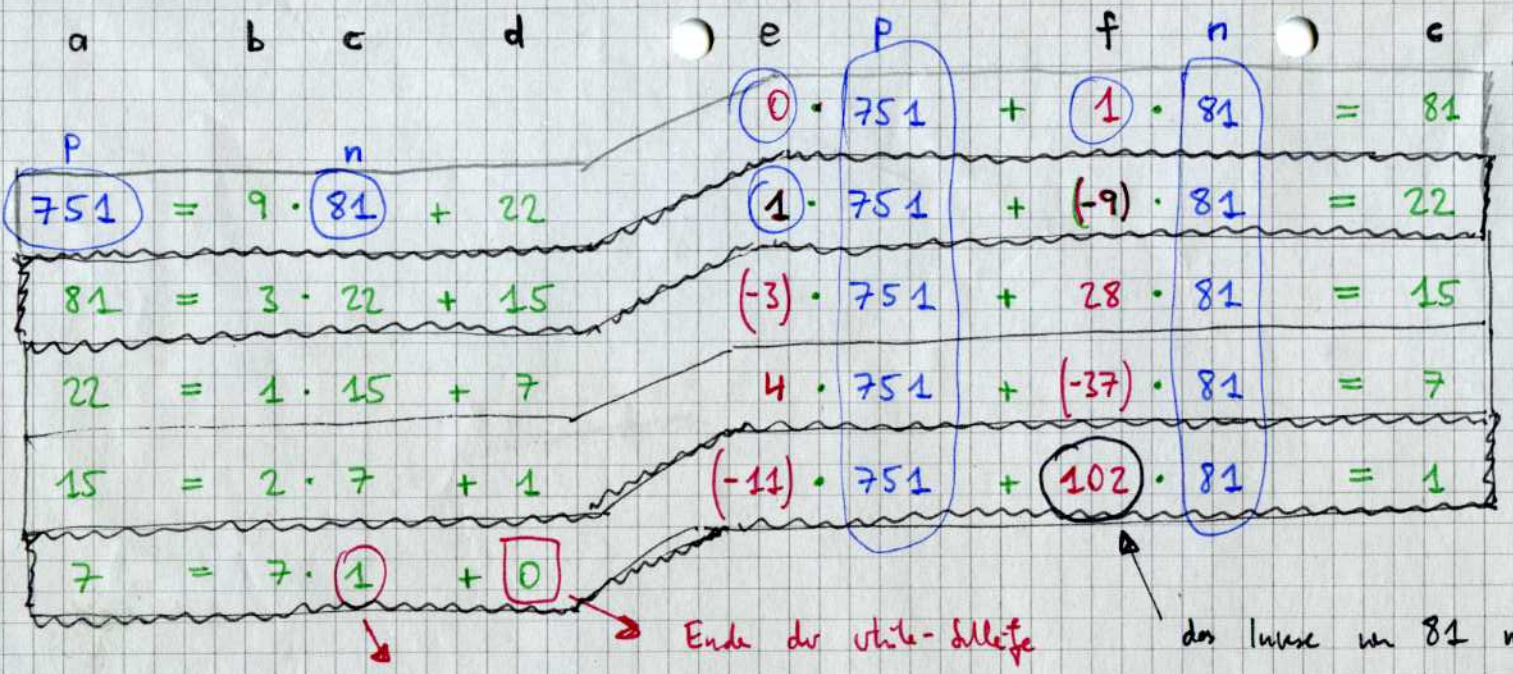
$$53 = 1 \cdot 39 + 14$$
$$39 = 2 \cdot 14 + 11$$
$$14 = 1 \cdot 11 + 3$$
$$11 = 3 \cdot 3 + 2$$
$$3 = 1 \cdot 2 + (1)$$
$$2 = 2 \cdot (1) + 0 \quad \text{stop}$$

$\Rightarrow$

$$1 = 3 - 1 \cdot 2$$
$$= 3 - 1 \cdot (11 - 3 \cdot 3) =$$
$$= -1 \cdot 11 + 4 \cdot 3 =$$
$$= -1 \cdot 11 + 4 \cdot (14 - 1 \cdot 11)$$
$$= -5 \cdot 14 + 4 \cdot 14 =$$
$$= -5 \cdot (39 - 2 \cdot 14) + 4 \cdot 14$$
$$= -5 \cdot 39 + 14 \cdot 14 =$$
$$= -5 \cdot 39 + 14 \cdot (53 - 1 \cdot 39) =$$
$$= -19 \cdot 39 + 14 \cdot 53 \overset{!}{\equiv} -19 \cdot 39$$
$$\overset{!}{\equiv} 34 \cdot 39 \overset{!}{\equiv} 1$$

Kontrolle:
$34 \cdot 39 = 1326 = 25 \cdot 53 + 1$ ✓

Übersicht über die folgenden Blätter :



16a

$22 = 1 \cdot 751 - 9 \cdot 81$

$15 = 81 - 3 \cdot 22 = 81 - 3 \cdot (1 \cdot 751 - 9 \cdot 81) = (-3 \cdot 1 + 0) \cdot 751 + (1 + (-3)(-9)) \cdot 81$

$= -3 \cdot 751 + 28 \cdot 81$

$7 = 22 - 1 \cdot 15 = (1 \cdot 751 - 9 \cdot 81) - 1 \cdot (-3 \cdot 751 + 28 \cdot 81) =$

$= [1 + (-1) \cdot (-3)] \cdot 751 + [-9 + (-1) \cdot 28] \cdot 81 = 4 \cdot 751 - 37 \cdot 81$

man braucht immer die letzten beiden Zeilen !!

$1 = 15 - 2 \cdot 7 = (-3 \cdot 751 + 28 \cdot 81) - 2 \cdot (4 \cdot 751 - 37 \cdot 81) =$

$= [-3 + (-2) \cdot 4] \cdot 751 + [28 + (-2) \cdot (-37)] \cdot 81 = -11 \cdot 751 + 102 \cdot 81$

Wie geht es mit den neuen Einträgen für e ? und für f ?

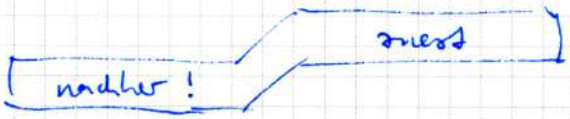
$e_{neu} = e(vor\text{-}vorher) + (-b) \cdot (e_{vorher})$

$f_{neu} = f_{vor\text{-}vorher} + (-b) \cdot f_{vorher}$!

Wir müssen dabei die 1. Zeile separat ausfüllen.

Bei der 2. Zeile kommt es richtig, wenn wir zunächst für $e_{vorher} = 0$ und $f_{vorher} = 1$ setzen !!, also eine 'nullte' Hilfs-Zeile ergänzen !

Reihenfolge: e, f, a, c, b, d !!



Delphi - Pascal
oder
Turbo - Delphi

```
39: procedure TForm1.Button1Click(Sender: TObject);
40: var n,p,a,b,c,d,e0,e1,e2,f0,f1,f2 : Integer ;
41:     s : string ;
42: begin
43:   n := StrToInt(Edit1.Text);
44:   p := StrToInt(Edit2.Text);
45:   if p = 0 then
46:     begin
47:       s := 'Geben Sie doch einen vernünftigen Wert ein für p !';
48:       Memo1.Text := s ;
49:       exit
50:     end ;
51:
52:   if n = 0 then
53:     begin
54:       s := '0  hat kein Inverses, mein Lieber !';
55:       Memo1.Text := s ;
56:       exit
57:     end ;
58:
59:   if (n = 1) or (n = -1) then
60:     begin
61:       s := 'Das Inverse von 1 ist 1 , dasjenige von -1 ist -1 ...'
62:       Memo1.Text := s ;
63:       exit
64:     end ;
65:
66:   a := n ;
67:   c := p mod n ;
68:
69:   if c = 0 then
70:     begin
71:       s := 'Das Inverse existiert nicht.'+chr(10)+chr(13);
72:       s := s + 'p ist durch n teilbar !';
73:       Memo1.Text := s ;
74:       exit
75:     end ;
76:
77:   b := a div c ;
78:   d := a mod c ;
79:   e0 := 0 ;
80:   e1 := 1 ;
81:   f0 := 1 ;
82:   f1 := - (p div n) ;
83:
84:   while not(d = 0) do
85:     begin
86:       e2 := e0 - b*e1 ;
87:       e0 := e1 ;
88:       e1 := e2 ;
89:       f2 := f0 - b*f1 ;
90:       f0 := f1 ;
91:       f1 := f2 ;
92:
93:       a := c ;
94:       c := d ;
95:       b := a div c ;
96:       d := a mod c
97:     end ;
98:
99:   if c = 1 then    // das Inverse existiert !
100:    begin
101:      while f1 < 0 do f1 := f1 + p ;
102:      f1 := f1 mod p ;
103:      s := 'Das Inverse von n modulo p existiert.' ;
104:      s := s + chr(13) + chr(10) + chr(13) + chr(10) ;
105:      s := s + 'Es beträgt      ' + IntToStr(f1) ;
106:      Memo1.Text := s ;
107:    end
108:  else
109:    begin
110:      s := 'Das Inverse von n modulo p existiert nicht.' ;
111:      s := s + chr(13) + chr(10) + chr(13) + chr(10) ;
112:      s := s + 'Der ggT der beiden Zahlen ist  ' + IntToStr(c) ;
113:      Memo1.Text := s ;
114:    end ;
115:
116: end;
```

Erweiterter Euklidischer Algorithmus

OH 2008 / Gub

"Probieren in for-Schleife"

```
(n,p)
Func
Local i
mod(n,p)→n
For i,1,p-1,1
  If mod(i*n,p)=1 Then
     Exit
  EndIf
EndFor
Return mod(i,p)
EndFunc
```

Funktion modinver(n, p)

Input: n Integer, p ∈ ℕ, man rechnet in $\mathbb{Z}_p$

Output: $\bar{n}^{-1}$ mit $\bar{n} \cdot n \equiv 1 \mod p$, falls $\bar{n}$ existiert
0 sonst

Wird schnell langsamer für größere Werte von p. Schon für p = 751 ist euklidin(n, p) viel schneller!

```
(a,b)
Func
Return (a-mod(a,b))/b
EndFunc
```

TI- Rechner !

Funktion div (a, b)

Input : a, b ∈ ℤ

Output : a div b

(genauer : [a - (a mod b)] div b

= [a - mod(a, b)] / b)

Nützliche Hilfsfunktion beim Rechnen in $\mathbb{Z}_p$.
Wird benutzt von der Funktion euklidin(..., ...)

mod(a, b) ist vordefiniert und liefert a modulo b

also mod(15, 4) = 3