

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

Für jedes $n \in \mathbb{Z}_p$ mit $0 < n < p$ ist
 $\text{ggT}(n, p) = 1$
 wenn p eine Primzahl ist.

Der euklid. Algorithmus und also der ggT 1 liefern:

Bsp. 1

$$17 = 3 \cdot 5 + 2$$

$$5 = 2 \cdot 2 + \textcircled{1}$$

$$2 = 2 \cdot \textcircled{1} + \underline{0} \text{ stop}$$

ggT, obwohl mit Wert $\neq 0$

$$\begin{aligned} \text{also: } 1 &= 5 - 2 \cdot 2 \\ &= 5 - 2(17 - 3 \cdot 5) = \\ &= 5 + 6 \cdot 5 - 2 \cdot 17 = \\ &= 7 \cdot 5 - 2 \cdot 17 \stackrel{!}{=} 7 \cdot 5 \pmod{17}! \end{aligned}$$

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

Bsp. 2

$$39^{-1} \text{ in } \mathbb{Z}_{53} \quad ?? \quad 53 \text{ ist prim!}$$

isPrime(53) ✓

$$53 = 1 \cdot 39 + 14$$

$$39 = 2 \cdot 14 + 11$$

$$14 = 1 \cdot 11 + 3$$

$$11 = 3 \cdot 3 + 2$$

$$3 = 1 \cdot 2 + \textcircled{1} \quad \Rightarrow$$

$$2 = 2 \cdot \textcircled{1} + \underline{0} \text{ stop}$$

$$1 = 3 - 1 \cdot 2$$

$$= 3 - 1 \cdot (11 - 3 \cdot 3) =$$

$$= -1 \cdot 11 + 4 \cdot 3 =$$

$$= -1 \cdot 11 + 4 \cdot (14 - 1 \cdot 11) =$$

$$= -5 \cdot 11 + 4 \cdot 14 =$$

$$= -5 \cdot (39 - 2 \cdot 14) + 4 \cdot 14 =$$

$$= -5 \cdot 39 + 14 \cdot 14 =$$

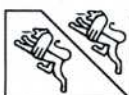
$$= -5 \cdot 39 + 14 \cdot (53 - 1 \cdot 39) =$$

$$= -29 \cdot 39 + 14 \cdot 53 \stackrel{!}{=} -29 \cdot 39$$

$$\stackrel{!}{=} \underline{\underline{34 \cdot 39}} \stackrel{!}{=} 1$$

Kontrolle:

$$34 \cdot 39 = 1326 = 25 \cdot 53 + 1$$



Übersicht über die folgenden Blätter:

- 16a Detailliertes Beispiel für die Sandtafel
- 16b Verallgemeinerung von 16a, Überlegungen für die Implementierung mit Excel
- 16c Implementierung mit Delphi 6.0 oder Turbo Delphi
- 16d TI-Rechner: Probieren mit for-Schleife
- 16e TI-TR: Die Hilfsfunktion $\text{div}(a, b)$
- 16f TI-TR: Erweiterter euklid. Algorithmus,
→ Funktion $\text{euklidin}(a, p)$
- 16g wie 16f, für den neuen TI-nSpire
- 16h ~~Die~~ weitere, kleine Funktion für die TI-Rechner:
 $\text{nextprim}(n)$
- 16i Syntax-Vergleich "Delphi-Pascal" ~ "TI-Bascal"

Variablenname !

Ergänzung ...

Start, Verkürzung, Stücke

Input !

Ende der Stücke - Stücke

a	b	c	d	e	f	n	e
751	$9 \cdot 81$		$+ 22$	$0 \cdot 751$	$+ 1 \cdot 81$		$= 81$
81	$3 \cdot 22$		$+ 15$	$1 \cdot 751$	$+ (-9) \cdot 81$		$= 22$
22	$4 \cdot 15$		$+ 7$	$(-3) \cdot 751$	$+ 28 \cdot 81$		$= 15$
15	$2 \cdot 7$		$+ 1$	$4 \cdot 751$	$+ (-37) \cdot 81$		$= 7$
7	$7 \cdot 1$		$+ 0$	$(-14) \cdot 751$	$+ 102 \cdot 81$		$= 1$

Ende der Stücke - Stücke

falls 1 existiert das keine !

das keine von 81 mod 751 (falls e=1)

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

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

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

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

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

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

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

man braucht immer die letzten beiden Zahlen !!

Wie gerinnt man die neuen Einträge für e ? und für f ?

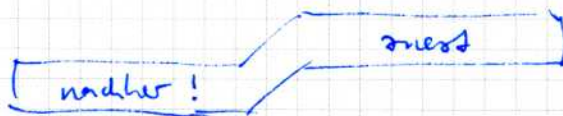
$$e_{\text{neu}} = e(\text{vor-vorher}) + (-b) \cdot (e_{\text{vorher}})$$

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

Wir müssen dabei die 1. Zeile separat einfüllen.

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

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



```

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;

```

Delphi - Pascal
oder
Turbo - Delphi

OLA 2008 / Gub

Algorithmen

Euklidischer

Erweiteter

"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, $p \in \mathbb{N}$, man reduziert in \mathbb{Z}_p

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

Wird schnell langsamer für größere Werte von p . Schon für $p=751$ ist $\text{modin}(n,p)$ viel schneller!

```

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

```

TI-Rechner!

Funktion $\text{div}(a, b)$ Input: a, b sind IntegerOutput: $a \text{ div } b$

$$\left(\text{genau: } \begin{aligned} & [a - (a \bmod b)] \text{ div } b \\ & = [a - \text{mod}(a, b)] / b \end{aligned} \right)$$

Wichtige Hilfsfunktion beim Reduzieren in \mathbb{Z}_p .
 Wird benutzt um die Funktion $\text{euclid}(\dots, \dots)$

$\text{mod}(a, b)$ ist undefiniert und liefert a modulo b
 also $\text{mod}(15, 4) = 3$ ✓✓

