Euklidischer Algorithmus

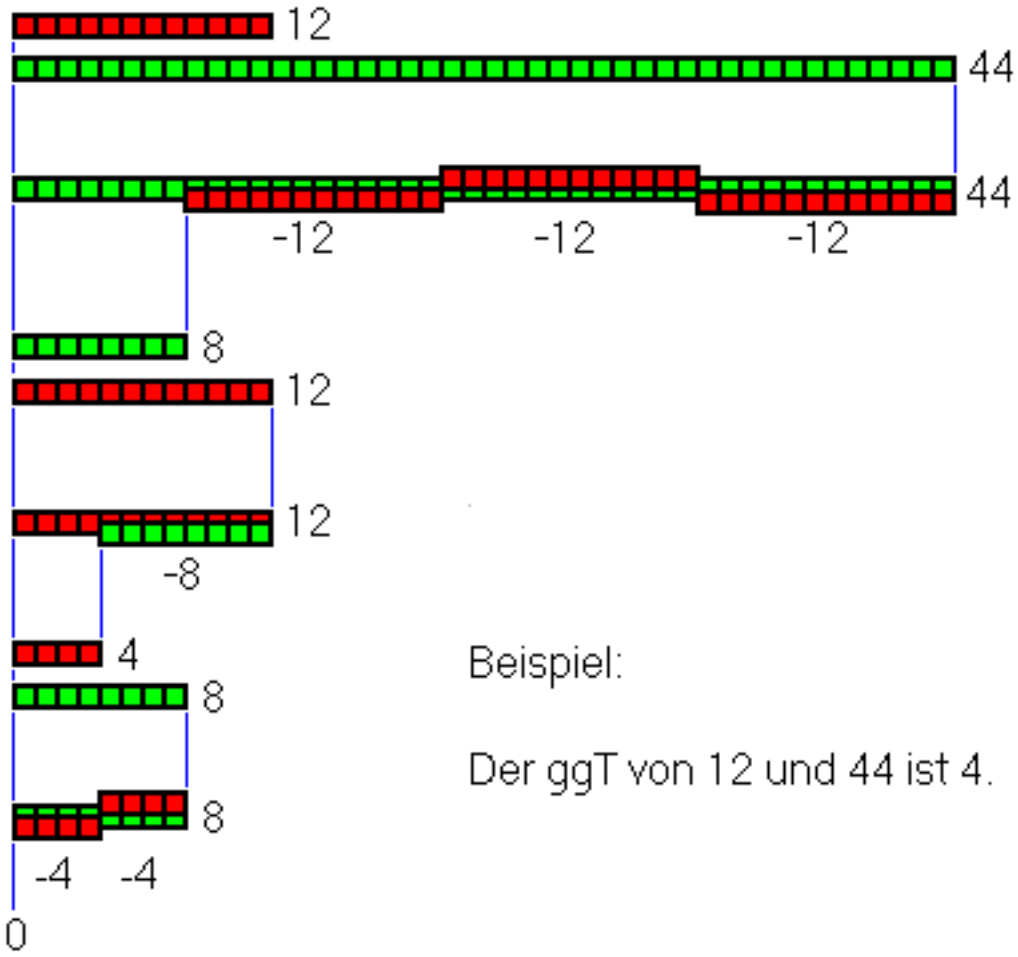
Wenn CD aber AB nicht misst, und man nimmt bei AB, CD abwechselnd immer das kleinere vom größeren weg, dann muss schließlich eine Zahl übrig bleiben, die die vorangehende misst.

(Aus Euklid, **Die Elemente**, Herausgegeben von Clemens Thaer, Wissenschaftliche Buchgesellschaft Darmstadt, VII Buch, §2)

Euklid berechnete den größten gemeinsamen Teiler, indem er nach einem gemeinsamen „Maß“ für die Längen zweier Linien suchte.

Dazu zog er wiederholt die kleinere der beiden Längen von der größeren ab.

Dabei nutzt er aus, dass sich der größte gemeinsame Teiler zweier Zahlen (oder Längen) nicht ändert, wenn man die kleinere von der größeren abzieht.



Heute ersetzt man die im klassischen Algorithmus auftretenden wiederholten Subtraktionen eines Wertes jeweils durch eine einzige **Division mit Rest**.

Der moderne euklidische Algorithmus führt nun in jedem Schritt solch eine Division mit Rest aus. Er beginnt mit den beiden Zahlen **a** und $\mathbf{b=r_0}$, deren größter gemeinsamer Teiler bestimmt werden soll.

$$\mathbf{a = q_1 * r_0 + r_1}$$

In jedem weiteren Schritt wird mit Disvisor und Rest des vorhergehenden Schritts eine erneute Division mit Rest durchgeführt. So lange, bis eine Division aufgeht, d.h. der Rest Null ist.

$$\mathbf{r_0 = q_2 * r_1 + r_2}$$
$$\mathbf{r_1 = q_3 * r_2 + r_3}$$
…
$$\mathbf{r_{n-1} = q_{n+1} * r_n + 0}$$

Der Divisor $\mathbf{r_n}$ der letzten Division ist der größte gemeinsame Teiler.

$$\mathbf{ggT(a, b) = r_n}$$

Da sich die Zahlen in jedem Schritt mindestens halbieren, ist der Algorithmus auch bei großen Zahlen sehr schnell.

Beispiel

ggT(1071, 1029) = ggT(1029, 1071 % 1029) = ggT(1029, 42) =

ggT(42, 1029 % 42) = ggT(42, 21) = ggT(21, 42 % 21) =

ggT(21, 0) = 21

1071 = 1 * 1029 + 42
1029 = 24 * 42 + 21
42 = 2 * 21 + 0

```c
#include <stdio.h>              // EuklidAlg.c
#include <math.h>
#define N 128

long ggT0(long a, long b) {      // rekursiv, modern
       if (!b)
              return abs(a);
       return abs(ggT0(b, a%b));
}

long ggT(long a, long b) {
       return b ? abs(ggT(b, a%b)) : abs(a);
}

// Der ggt(n,m) wird m, wenn n durch m teilbar, anderenfalls ist er
// gleich dem ggt(m, Rest(n/m))
long ggT1(long n, long m) {      // rekursiv
      if(abs(m)>abs(n)) return abs(ggT1(m,n));
      if(!m) return abs(n);
      return abs(ggT1(m,n%m));
}

// ggT(6,15) -> ggT(15,6) -> ggT(6,15%6) -> ggT(6,3) -> ggT(3,0) -> 3
//     n  m          n m         n    m         n m         n m   ggt

long ggT2(long a, long b) {      // rekursiv old
       a = abs(a); b = abs(b);
       if (!b) return a;
       if (!a) return b;
       if (a > b) return ggT2(a - b, b);
       return ggT2(a, b - a);
}

long Euklid0(long a, long b) {
       long h;
       a = abs(a); b = abs(b);
       while (b)
              h = a%b, a = b, b = h;
       return a;
}

long Euklid1(long a, long b) {    // iterativ
       a = abs(a); b=abs(b);
       while(a*b){
                     if(a >= b) a = a % b;
                        else b = b% a;
       }
       return a + b;
}
```

```
long Euklid2(long a, long b) {   // iterativ old
	a = abs(a); b = abs(b);
	if (!a)
		return b;
	while (b) {
		if (a > b) a -= b;
		else b -= a;
	}
	return a;
}

void main(){
	int a = 0, b = 0, i = 0;
	char puffer[N], *s = 0;

	do {
		clearerr_s(stdin);
		printf("ganze Zahl a      =     ");
		s = fgets(puffer, N, stdin);
		if (s <= 0 || feof(stdin) || ferror(stdin)) continue;
		i = sscanf_s(puffer, "%5d", &a);
	} while (s <= 0 || feof(stdin) || ferror(stdin) || i <= 0);

	do {
		clearerr_s(stdin);
		printf("ganze Zahl b      =     ");
		s = fgets(puffer, N, stdin);
		if (s <= 0 || feof(stdin) || ferror(stdin)) continue;
		i = sscanf_s(puffer, "%5d", &b);
	} while (s <= 0 || feof(stdin) || ferror(stdin) || i <= 0);

	printf("ggT0( %d , %d )    = %5d\n", a, b, ggT0(a, b));
	printf("ggT( %d , %d )     = %5d\n", a, b, ggT(a, b));
	printf("ggT1( %d , %d )    = %5d\n", a, b, ggT1(a,b));
	printf("ggT2( %d , %d )    = %5d\n", a, b, ggT2(a, b));
	printf("Euklid0( %d , %d ) = %5d\n", a, b, Euklid0(a, b));
	printf("Euklid1( %d , %d ) = %5d\n", a, b, Euklid1(a,b));
	printf("Euklid2( %d , %d ) = %5d\n", a, b, Euklid2(a, b));
	getchar();
}

/*
ganze Zahl a      =     6
ganze Zahl b      =    15
ggT0( 6 , 15 )    =     3
ggT( 6 , 15 )     =     3
ggT1( 6 , 15 )    =     3
ggT2( 6 , 15 )    =     3
Euklid0( 6 , 15 ) =     3
Euklid1( 6 , 15 ) =     3
Euklid2( 6 , 15 ) =     3
*/
```