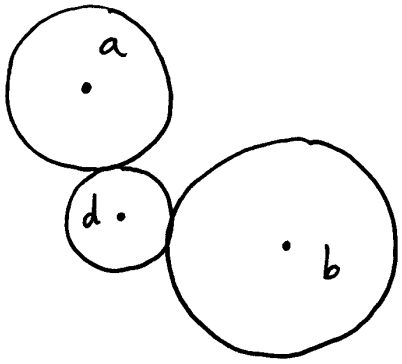
"Vom Problem zur Lösung": ein einfaches Beispiel

(aus "Einführung in die Informatik", Vorlesung, W. Lex, TU Clausthal, 1983)

A. Problem:
Bei einem Zahnradgetriebe soll das Antriebsrad je ein Rad mit *a*, etwa 105, und *b*, etwa 147, Zähnen derart treiben, dass einer vollen Umdrehung eines getriebenen Rades eine oder mehrere volle Umdrehungen des treibenden Rades entsprechen und die Übersetzungen möglichst klein sind. Wieviele Zähne muss das Antriebsrad haben?

B. Mathematisierung



Sei *d* die gefragte Zahl.
Dann gilt offenbar:
d teilt *a* und *d* teilt *b*.
Da die Übersetzungen minimal sein sollen, muss d maximal sein, also $d = ggT(a, b)$
(größter gemeinsamer Teiler).

C. Finden eines Lösungsalgorithmus:
Euklid (um 325 v. Chr., "Elemente", Buch 7, Problem 1, Proposition 2), wahrscheinlich schon bei Eudoxos von Knidos um 375 v. Chr.:
Euklidischer Algorithmus
147 : 105 → 1 Rest 42
105 : 42 → 2 Rest 21
42 : **21** → 2 Rest **0**
ggT(105; 147) = **21**

D. Formulierung des Algorithmus:
I. Falls $a \neq b$, sei *x* die größere und *y* die kleinere der Zahlen *a*, *b*;
bei $a = b$ sei $x = a = y$.
II. Berechne *x*:*y* mit Rest.
III. Ergibt sich kein Rest, ist $y = ggT(a, b)$.
IV. Ergibt sich ein Rest *r*, ersetze *x* durch *y* und *y* durch *r* und gehe nach II.

E. Präzisierung dieser Formulierung:

Seien a, b natürliche Zahlen und o.B.d.A. $b \leq a$, ferner $x_0 = a$, $x_1 = b$ und $x_{n-1} = q_n x_n + x_{n+1}$
mit $q_n, x_{n+1} \in \{0; 1; 2; ...\}$ und $x_{n+1} < |x_n|$ für $n > 0$ und $x_n \neq 0$.
Sei n der erste Index mit $x_{n+1} = 0$. Dann ist $x_n = \text{ggT}(a, b)$.

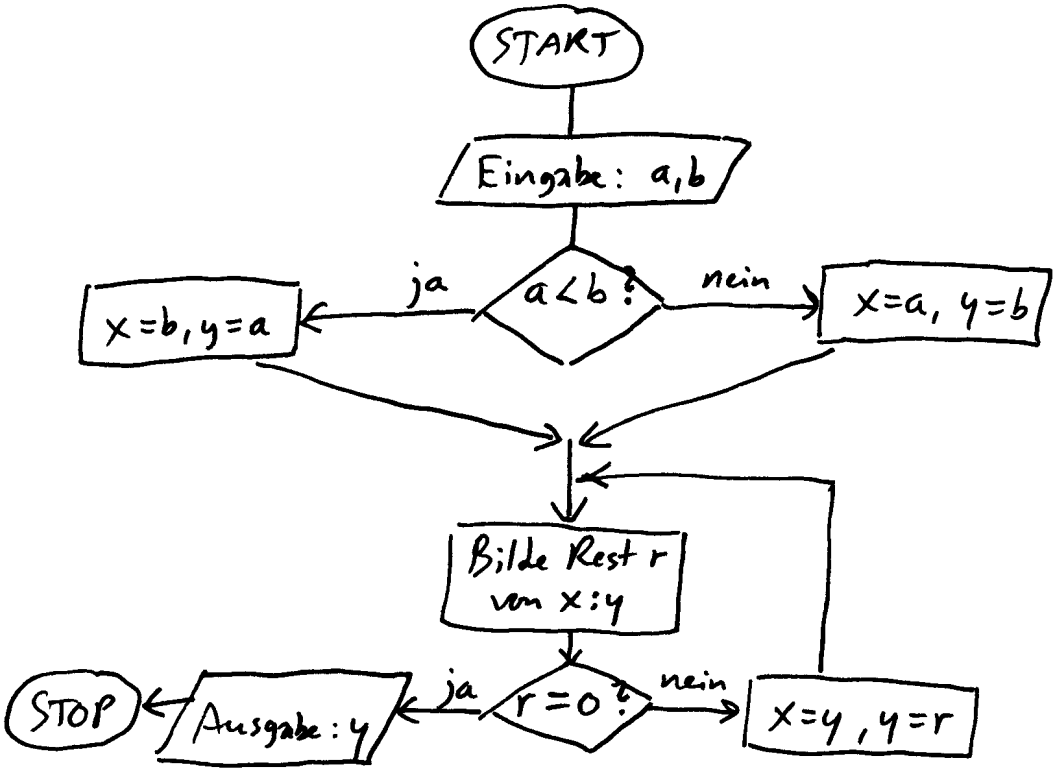
F. Beweis der Adäquatheit des Algorithmus
Zunächst zu zeigen: der Algorithmus in E terminiert, d.h. es existiert stets eine nat. Zahl n, so dass $x_{n+1} = 0$.
Dies gilt wegen $x_{n+1} < x_n$ und $x_{n+1} \geq 0$: Es gibt nur endlich viele natürliche Zahlen zwischen x_n und 0, also nur endlich viele Schritte.
Es gilt:
x_n teilt $q_n x_n + 0 = x_{n-1}$
x_n teilt $q_{n-1} x_{n-1} + x_n = x_{n-2}$
x_n teilt $q_{n-2} x_{n-2} + x_{n-1} = x_{n-3}$
...
mittels vollständiger Induktion folgt
x_n teilt $q_2 x_2 + x_3 = x_1 = b$
x_n teilt $q_1 x_1 + x_2 = x_0 = a$
somit teilt x_n die ganzen Zahlen a und b und damit auch ggT(a, b).
Andererseits ist der ggT ein Teiler von x_0 und x_1 und nach obiger Kette von Gleichungen auch von x_2, x_3, ..., x_{n-1}, x_n.
Also $x_n = \text{ggT}(a, b)$, q.e.d.

G. Grob-Programmierung:

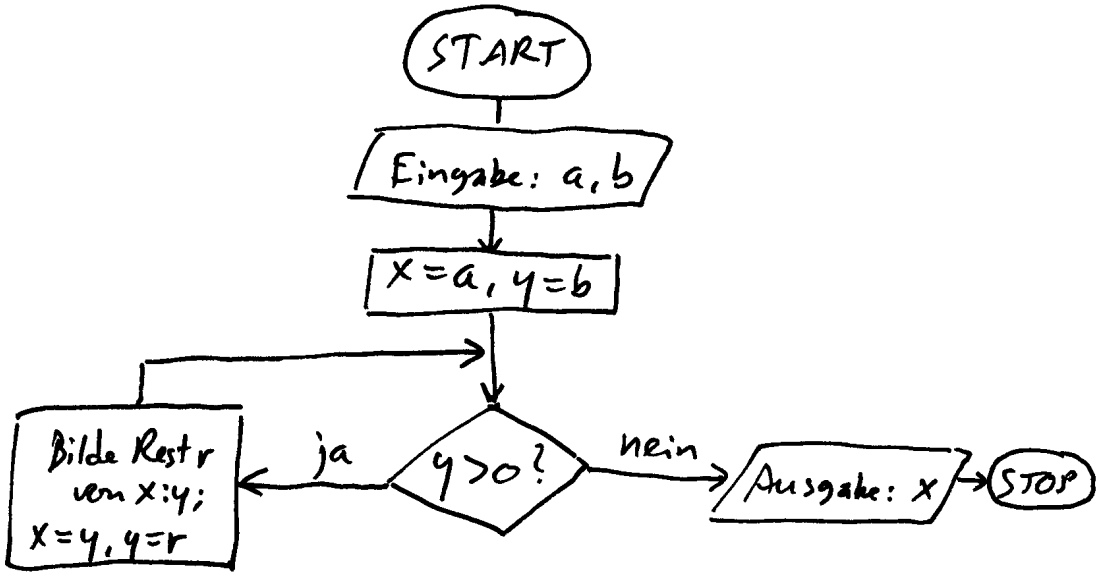
(a) Flussdiagramm (Programmablaufplan)

"Fluss", d.h. Programmablauf, wenn nicht anders durch Pfeile spezifiziert von oben nach unten und von links nach rechts.

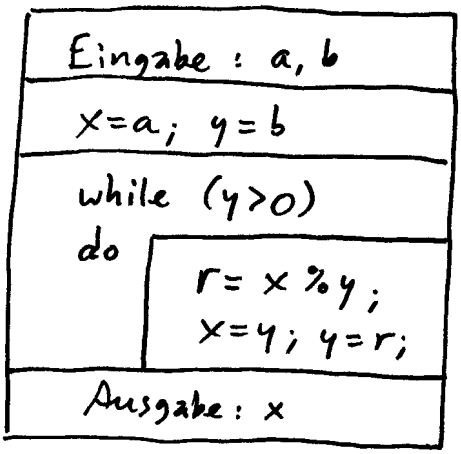
Erster Entwurf:



Der euklidische Algorithmus arbeitet auch im Fall $b \leq a$, daher Vereinfachung zu:



(b) Struktogramm (Nassi-Shneidermann-Diagramm),
dem zweiten Flussdiagramm entsprechend:



H. Auswahl der Programmiersprache und des Rechners:

C auf Intel-PC unter Windows

I. Feinprogrammierung:

```
#include <stdio.h>
int main()
   {
   char inbuf[50];
   int x, y, r;
   printf("\nBitte 2 Eingabezahlen: ");
   gets(inbuf);
   sscanf(inbuf, "%d %d", &x, &y);
   while (y>0)
      {
      r = x % y;
      x = y;
      y = r;
      }
   printf("\nErgebnis: %d\n", x);
   return 0;
   }
```

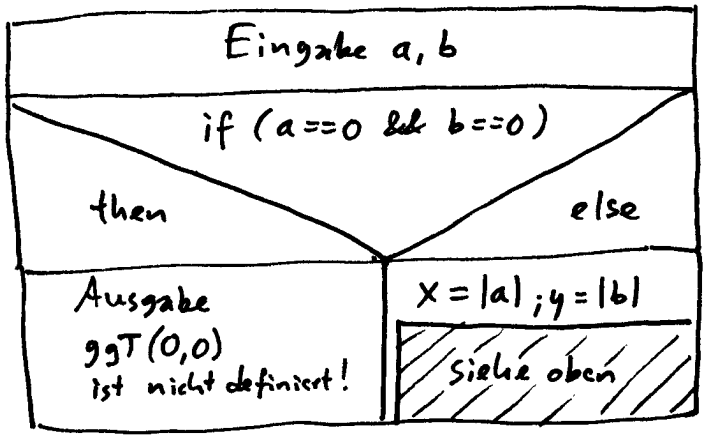
J. Programmverifikation:
Prüfen des Programms auf Korrektheit, d.h. auf Leistung des Gewünschten, indem man das gesamte Programm syntaktisch (d.h. den formalen Vorschriften entsprechend) kontrolliert und insbes. die einzelnen Variablenmanipulationen genau verfolgt.
In unserem Fall:
Verifikationszusicherungen
- nach Zeile 8: x, y natürliche Zahlen (falls korrekt eingelesen!)
- nach Zeile 11: $x = qy+r$, $q \geq 0$, $r \in \{0; 1; ...; y-1\}$
- nach Zeile 12: $x = y$
- nach Zeile 13: $y = r$
- nach Zeile 14: $y = 0$

man stellt fest, dass diese bei jedem möglichen Programmablauf erfüllt bleiben

K. Testen des Programms
Test durch bekannte Zahlenbeispiele; möglichst so, dass sämtliche denkbaren Alternativen erfasst werden. Dabei Extremfälle (etwa: etwas ist = 0) beachten.
Beispiele:
30 40
100 100
0 100
100 0
100 1

L. Rechnen
Eingabe 105 147 → Ausgabe 21,
das gesuchte Antriebsrad hat 21 Zähne.

M. Eventuelle Verallgemeinerungen, z.B. ggT für ganze Zahlen (bisherige Einschränkung auf nichtnegative Zahlen fallenlassen):



N. Eventuell Schreiben als Unterprogramm
(Funktion in C):

```
int ggT(int x, int y)
   {
   int r;
   while (y > 0)
      {
      r = x % y;
      x = y;
      y = r;
      }
   return x;
   }
```

diese kann in einem größeren Programm benutzt werden – etwa iterativ um den ggT von mehr als 2 Zahlen auszurechnen.

Aufruf z.B.: `z = ggT(p, q);`

....