

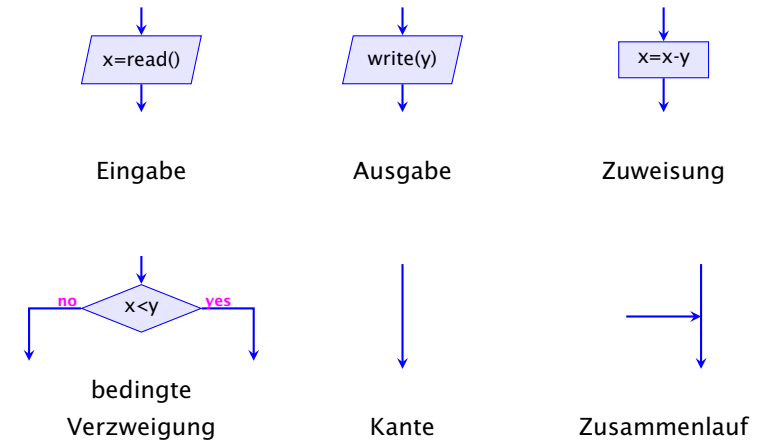
4 Kontrollflussdiagramme

In welcher Weise Programmteile nacheinander ausgeführt werden kann anschaulich durch **Kontrollflussdiagramme** dargestellt werden.

Zutaten:



4 Kontrollflussdiagramme

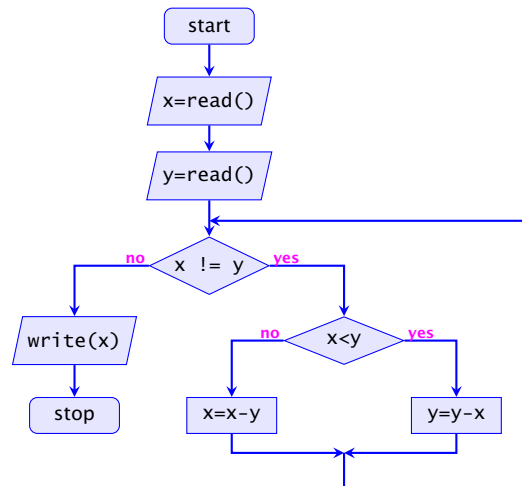


4 Kontrollflussdiagramme

Beispiel:

```
int x, y;
x = read();
y = read();
while (x != y) {
    if (x < y)
        y = y - x;
    else
        x = x - y;
}
write(x);
```

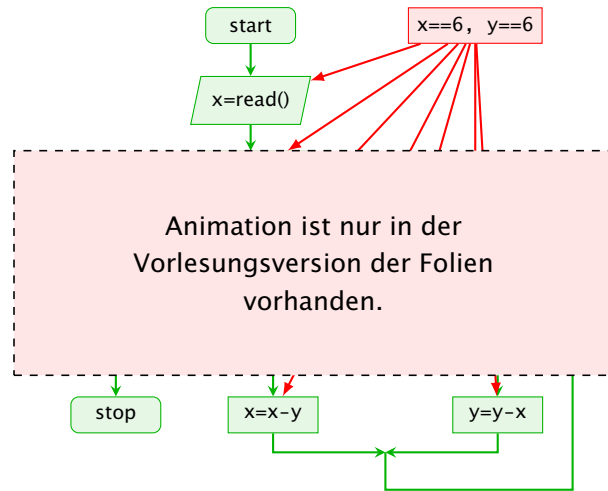
GGT



4 Kontrollflussdiagramme

- ▶ Die Ausführung des Programms entspricht einem **Pfad** durch das Kontrollflussdiagramm vom Startknoten zum Endknoten.
- ▶ Die Deklaration von Variablen muss man sich am Startknoten vorstellen.
- ▶ Die auf dem Pfad liegenden Knoten (außer Start- und Endknoten) sind Operationen bzw. auszuwertende Bedingungen.
- ▶ Um den Nachfolger an einem Verzweigungsknoten zu bestimmen, muss die Bedingung mit den aktuellen Werten der Variablen ausgewertet werden. (**operationelle Semantik**)

4 Kontrollflussdiagramme



4 Kontrollflussdiagramme

- ▶ zu jedem **Minijava**-Programm lasst sich ein Kontrollflussdiagramm konstruieren;
- ▶ die Umkehrung gilt auch, liegt aber nicht sofort auf der Hand

Die Umkehrung ware sehr einfach zu bewerkstelligen, wenn wir in einem Minijava-Programm **goto**-Befehle benutzen durften, d.h. wenn wir von jedem Punkt zu jedem anderen innerhalb des Programms springen konnten. Die obige Aussage bedeutet im Prinzip, dass man **goto**-Befehle immer durch geeignete Schleifen ersetzen kann.

4 Kontrollflussdiagramme

