Softwareentwicklung 1
UE-Stunde 01 - Algorithmen

Dr. Herbert Prähofer
Institut für Systemwissenschaften
Johannes Kepler Universität Linz

Worum geht es?

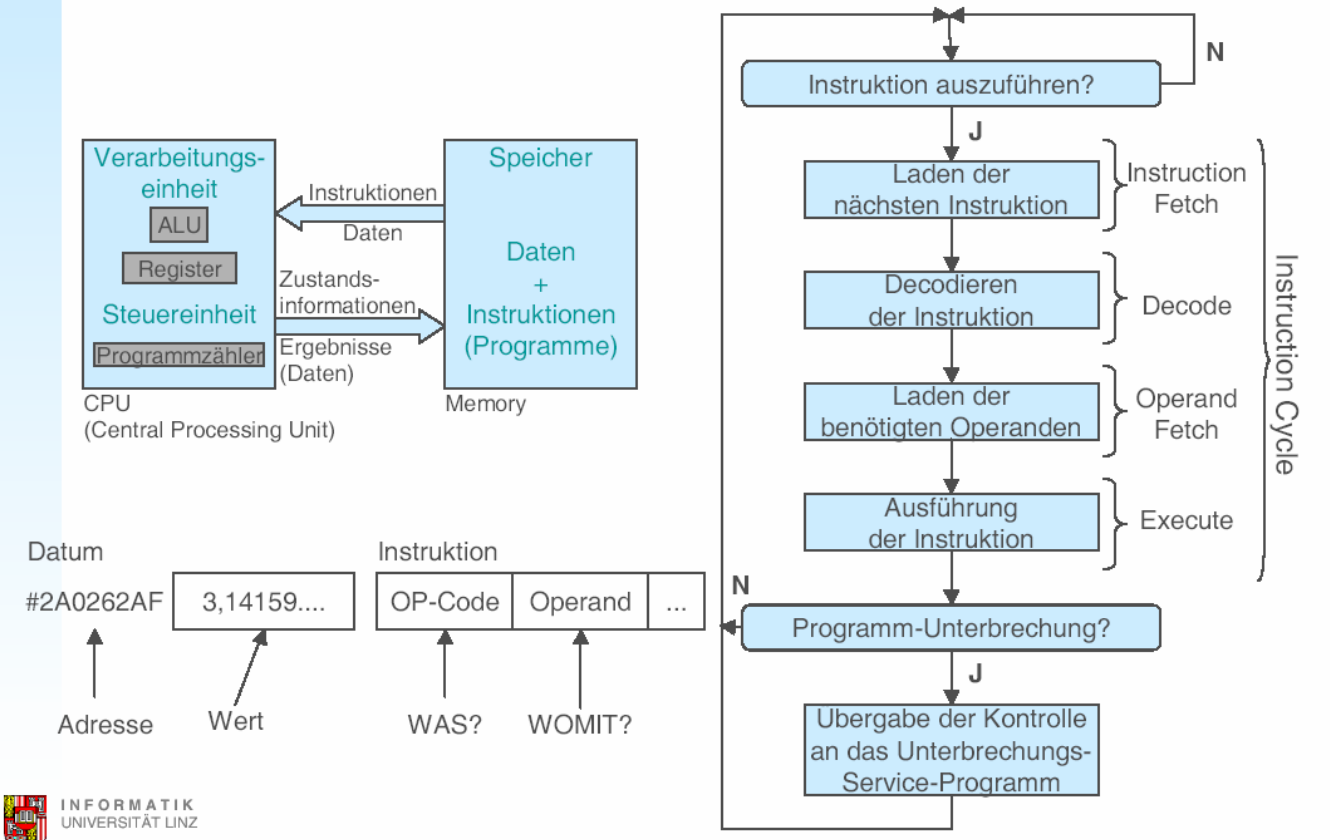
Programmieren

ein Problem so exakt beschreiben, dass es ein Computer lösen kann

- ☞ kreative Tätigkeit
- ☞ Ingenieurtätigkeit
- ☞ Nur wenige Leute können gut programmieren

Programm = Daten + Befehle

von Neumann Architektur



aus: A. Ferscha, Skriptum Softwareentwicklung 1, JKU Linz

RISC Programm

(hypothetischer) RISC Befehlssatz: Suchen in einer verketteten Liste

- R1 enthält Adresse des Listenanfanges
- R2 enthält Suchmuster

```
        MOVE    W  R1, R0      Laufvariable R0 initialisieren
loop:   CMP     W  I 0, R0     Testen ob "NULL" (Liste leer)
        JEQ        nein        fertig (Listenende), nicht gefunden
        CMP     W  R2, 8+!R0       Vergleich des Suchmusters mit Nutzdaten
        JEQ        ja          fertig, gefunden
        MOVE    W  !R0, R0     dereferenzieren
        JUMP       loop        Schleife
ja:     ...
nein:   ...
```

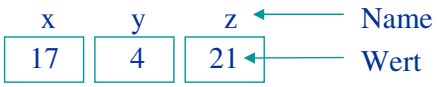

aus: A. Ferscha, Skriptum Softwareentwicklung 1, JKU Linz

Typische Befehle

- **`load:`** Laden von Daten aus Hauptspeicher
- **`store`**: Schreiben von Daten in Hauptspeicher
- **`move`**: Speicheroperationen
- **`add, sub, mult, `**…: Arithmetsiche Operationen
- **`cmp`**: Vergleichsoperationen: Vergleich auf 0
- **`jmp`**, **`jeq`**; Sprünge und bedingte Sprünge im Ablauf
- …

Daten und Befehle

Daten Menge adressierbarer Speicherzellen



Daten sind binär gespeichert (z.B. 17 = 10001)
Binärspeicherung ist universell (Zahlen, Texte, Bilder, Ton, ...)
1 Byte = 8 Bit
1 Wort = 2 Byte (oder 4 Byte)
1 Doppelwort = 2 Worte

Befehle Operationen mit den Speicherzellen

Maschinensprache		*Hochsprache*
ACC ← x	// Lade Zelle x	z = x + y;
ACC ← ACC + y	// Addiere Zelle y	
z ← ACC	// Speichere Ergebnis in Zelle z	

Variablen

Programme arbeiten mit Variablen.
Sind benannte Behälter für Werte.

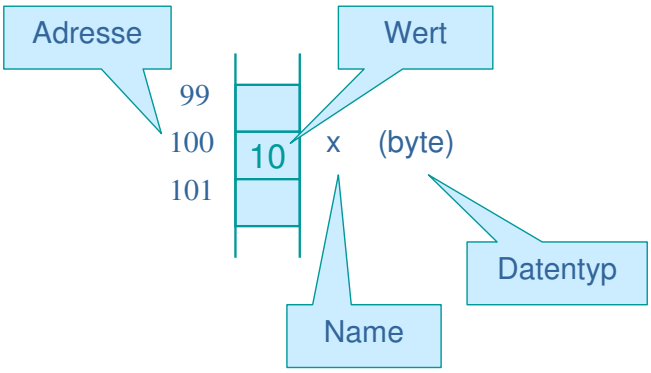
Variablen können ihren Wert ändern

$x \leftarrow x + 1$

Beispiel: `byte x = 10`

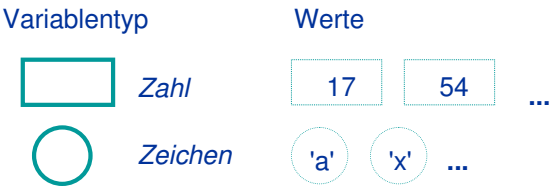
Charakterisiert durch

- Name oder Bezeichner
- Wert
- Adresse (im Hauptspeicher)
- Datentyp
  - int, char, byte, float, ...



Datentypen

Variablen haben einen Datentyp == Menge erlaubter Werte



Typ ≅ Form
- in eine Zahlenvariable passen nur Zahlen
- in eine Zeichenvariable passen nur Zeichen

Beispiele von Datentypen:

- char (1 Byte) [a-z][A-Z][0-9][Sonderzeichen]
- int (4 Byte) ~-2,147 Mrd. – ~+2,147 Mrd.
- boolean true, false
- float 3.14159265259

Algorithmus

Ein Algorithmus ist ein ***endliches***, ***schrittweises*** Verfahren zur ***Berechnung gesuchter aus gegebenen Größen***, in dem jeder Schritt aus einer Anzahl ***eindeutig*** ausführbarer Operationen und einer Angabe über den ***nächsten Schritt*** besteht.

- Berechnungsvorschrift
- schrittweise
- endlich
- ausführbare Operationen
- Angabe der Reihenfolge der Schritte

(Der Name *Algorithmus* ist abgeleitet von Al Chwarizmi, arabischer Mathematiker, ca. 800 n.Chr.)

Beispiel: Algorithmus „Sum“

Algorithmus „Sum“ zum Berechnen der Summe einer Zahlenfolge

Aufgabenstellung:

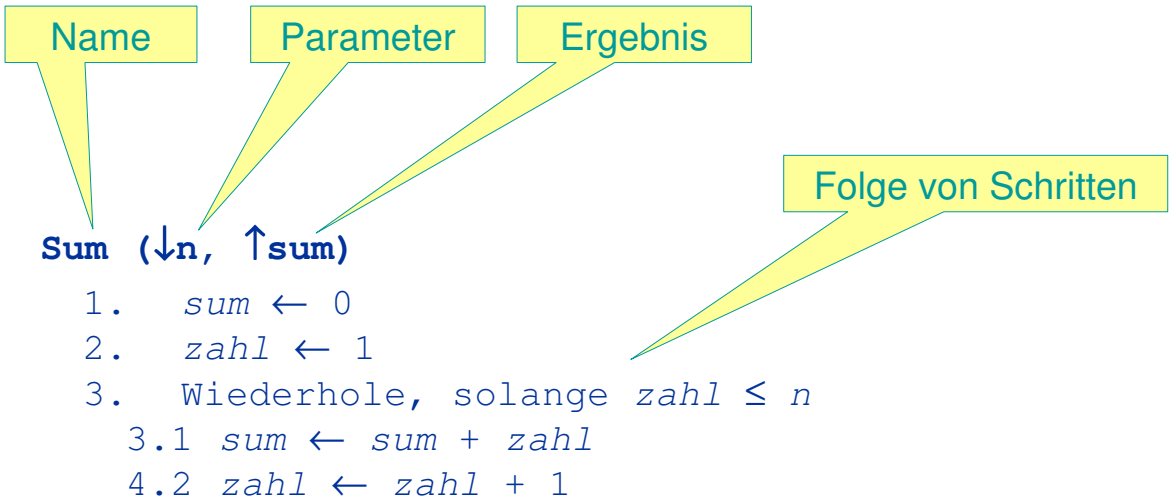
Gegeben: Eine Zahl n größer 0
Gesucht: Die Summe dieser Zahlen von 1 bis n

Algorithmus:

1. Addiere alle Zahlen von 1 bis n
2. Gib das Ergebnis aus

Diese Formulierung ist für einen Computer noch *zu wenig genau*!

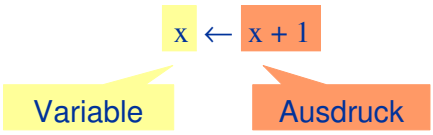
Algorithmus „Sum“ als Verfahren in mehreren Schritten



Diese Formulierung ist einem Computerprogramm schon *sehr nahe*.

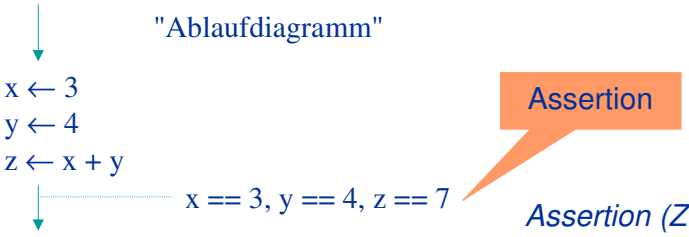
Anweisungen: Wertzuweisung und Sequenz

Wertzuweisung



1. werte Ausdruck aus
2. weise seinen Wert der Variablen zu

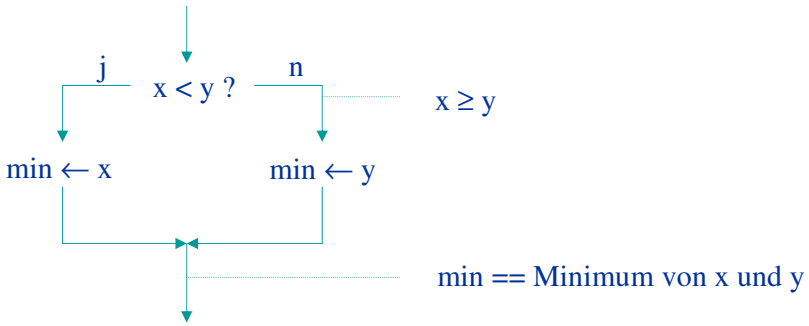
Anweisungsfolge (auch Sequenz)



Assertion (Zusicherung)
Aussage über den Zustand des Algorithmus an einer bestimmten Stelle

Anweisungen: Verzweigung

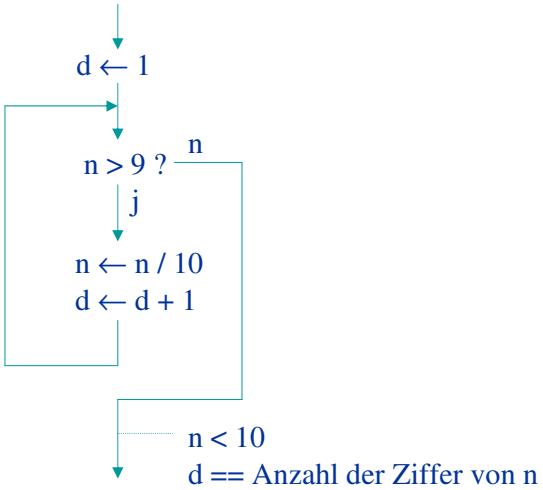
Auswahl (auch Verzweigung, Abfrage, Selektion)

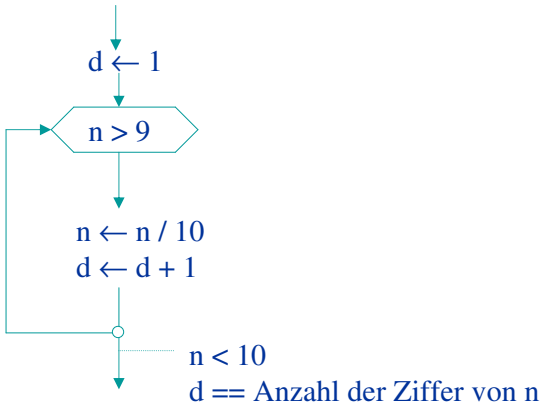


Anweisungen: Wiederholung

Wiederholung (auch Schleife, Iteration)

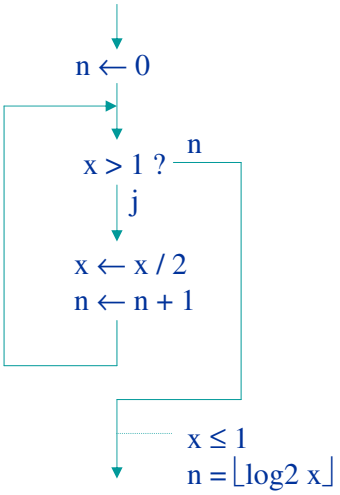
Alternative Darstellung

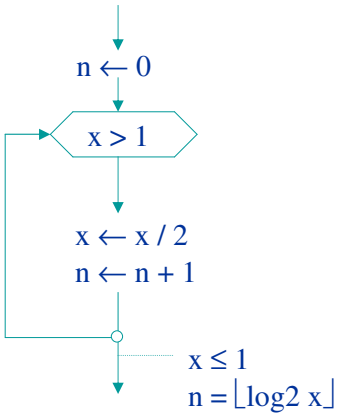




Anweisungen: Wiederholung

Wiederholung (auch Schleife, Iteration)





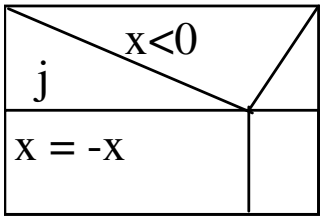
Alternative Darstellung

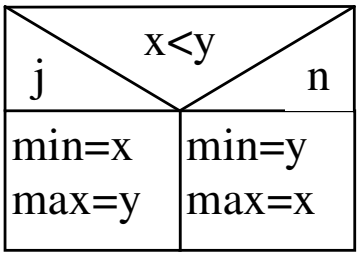
Struktogramm: Sequenz und Verzweigung

Sequenz

```
x=a+b
x= x+1
```

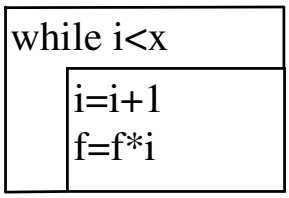
Verzweigung



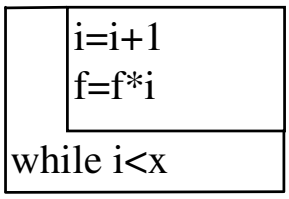


Struktogramm: Abweis-/Durchlaufschleife

Abweisschleife

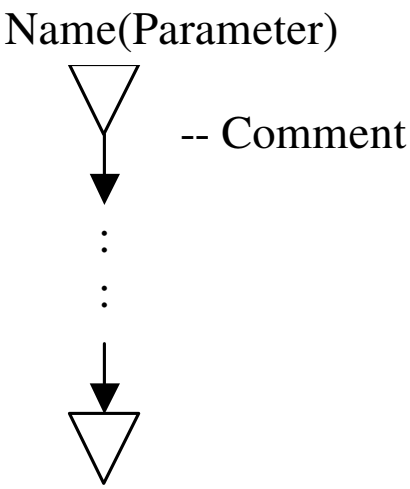


Durchlaufschleife

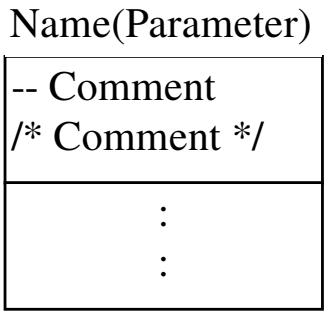


Programmanfang/-ende

Ablaufdiagramm

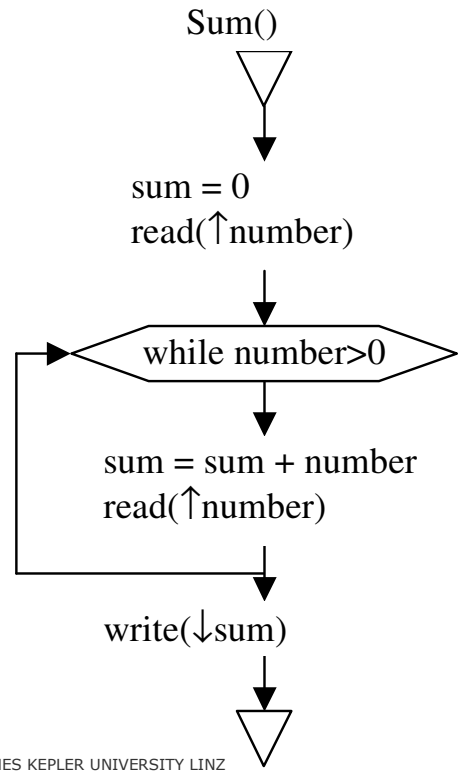


Struktogramm



Algorithmus "Sum"

Ablaufdiagramm

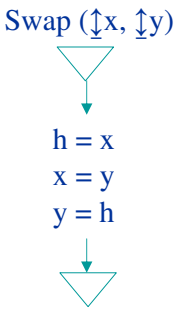


Struktogramm

Sum()

sum = 0
read(↑number)
while number>0
sum = sum + number
read(↑number)
write(↓number)

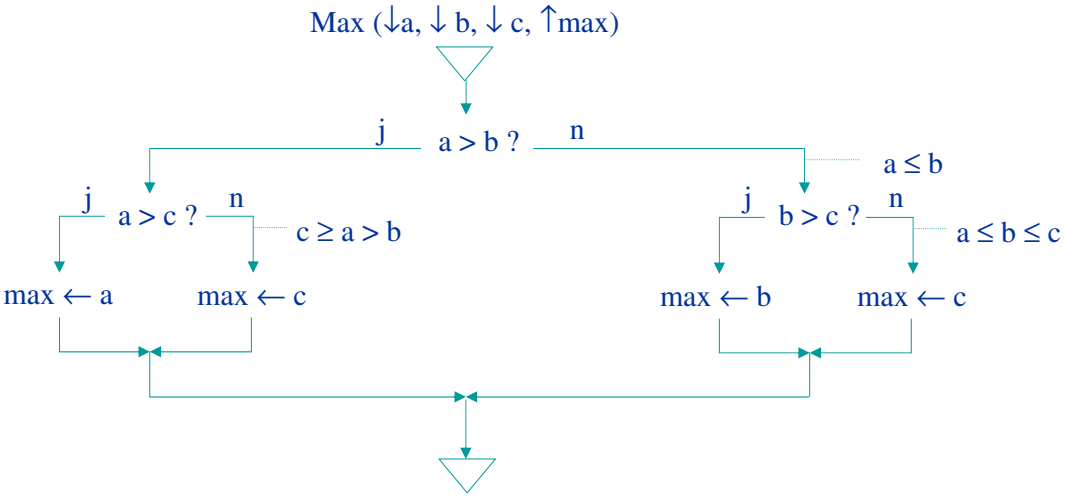
Beispiel: Vertauschen zweier Variableninhalte



Schreibtischtest

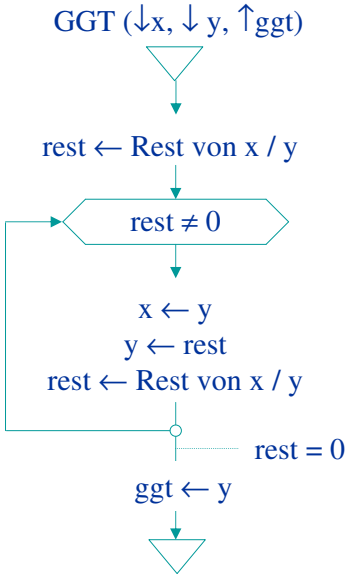
x	y	h
~~3~~	~~2~~	3
2	3	

Beispiel: Maximum dreier Zahlen bestimmen



Beispiel: Euklidscher Algorithmus

Berechnet den größten gemeinsamen Teiler zweier Zahlen *x* und *y*



Schreibtischtest

x	y	rest
~~28~~	~~20~~	8
~~20~~	~~8~~	4
8	4	0

Warum funktioniert dieser Algorithmus?
(ggt teilt x) & (ggt teilt y)
⇒ ggt teilt (x - y)
⇒ ggt teilt (x - q*y)
⇒ ggt teilt rest
⇒ GGT(x, y) = GGT(y, rest)

Algorithmenschreibweisen

Ablaufdiagramm (schon bekannt) ✓

Struktogramm

GGT (↓x, ↓ y, ↑ggt)

```
rest ← Rest von x / y
while rest ≠ 0
    x ← y
    y ← rest
    rest ← Rest von x / y
ggt ← y
```

\+ erzwingen strukturiertes Programmieren (ohne beliebige Sprünge)

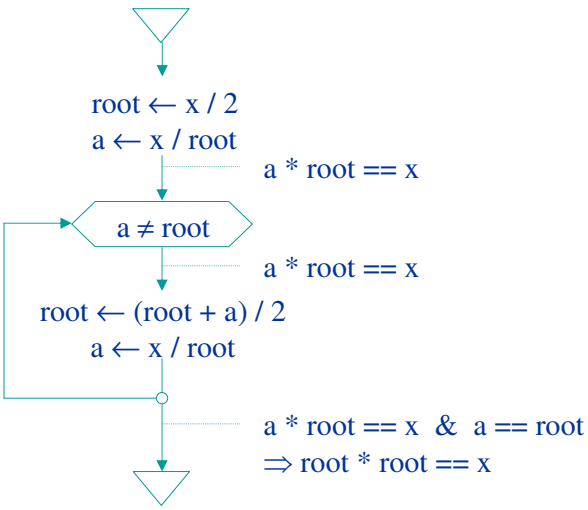
\- aufwendig zu zeichnen
schwer zu ändern

Beispiel: Quadratwurzel von x berechnen

Näherung:

root ← (a + root) / 2
a ← x / root

SquareRoot (↓x, ↑root)



Schreibtischtest

x	root	a
10	~~5~~	~~2~~
	~~3.5~~	~~2.85714~~
	~~3.17857~~	~~3.14607~~
	~~3.16232~~	~~3.16223~~
	3.16228	3.16228

Kommazahlen sind meist nicht exakt gleich, daher besser `|a - root| > 0.0000001`

Algorithmenschreibweisen

Stilisierte Prosa

GGT (↓x, ↓ y, ↑ggt)

S1 Bilde Rest.
Dividiere *x* durch *y* und nenne den Rest *rest*

S2 Ende?
Wenn *rest* null ist, gehe zu S4

S3 Ersetze.
Ersetze *x* durch *y* und *y* durch *rest*
Gehe nach S2

S4 Fertig.
Das Ergebnis *ggt* ist *y*

\+ größtmögliche Freiheit in der Formulierung

\- schreibaufwendig
\- unpräzise
\- Ablaufstrukturen (Schleifen, Verzweigungen) nicht sichtbar

Algorithmenschreibweisen

\+ vom Computer lesbar und verarbeitbar
\+ eindeutig und präzise
\+ von Menschen lesbar

\- exakt, ohne Freiheitsgrade
\- aufwendig zu schreiben

Java-Programm

```
public class GGTProgram {

  public static void main(String[] args) {

    int x;
    int y;
    int ggt;

    Out.print("Bitte x eingeben: ");
    x = In.readInt();
    Out.print("Bitte y eingeben: ");
    y = In.readInt();
    ggt = GGTEuklid(x, y);
    Out.print("Der GGT von x und y ist: ");
    Out.println(ggt);
  }

  static int GGTEuklid(int x, int y) {

    int rest;
    int ggt;

    // %-Operator bildet den Rest
    rest = x % y;
    while (rest != 0) {
      x = y;
      y = rest;
      rest = x % y;
    }

    // rest == 0
    ggt = y;
    return ggt;
  }
} // end GGTProgram
```

Algorithmen und Programmierung

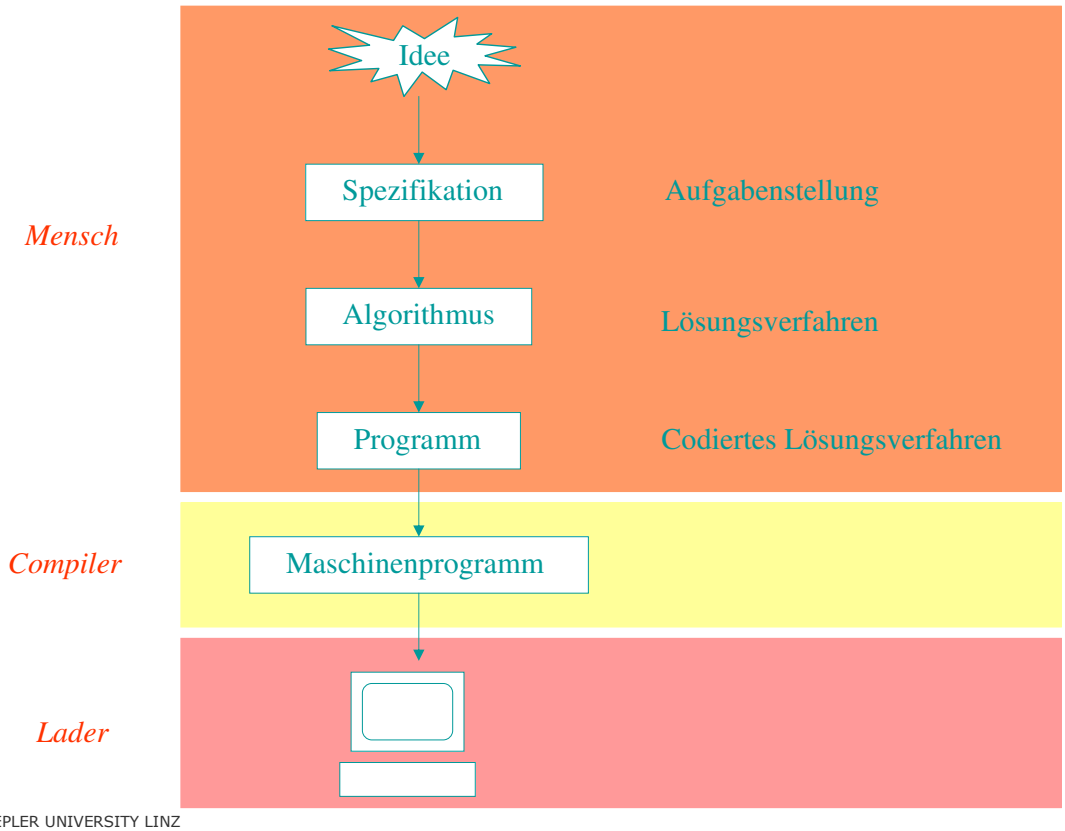
Programm

Ein für die Lösung einer bestimmten Aufgabe mit einer Datenverarbeitungsanlage geeigneter Algorithmus

d.h.

- Elementare Operationen sind die durch den Computer ausführbaren Befehle
- Programm muss in einer für den Computer lesbaren Form definiert sein (==> Programmiersprache)

Programmerstellung



Schritte des Algorithmenentwurfs und der Programmierung

- Problem erfassen und beschreiben
- Lösungsidee erarbeiten und niederschreiben
- Lösungsidee in einen schrittweisen Ablauf überführen (= Algorithmus)
- Algorithmus so weit verfeinern und konkretisieren, dass Umsetzung in Java direkt möglich ist
- Programmierung in Java
- Überlegen von Testfällen (dabei alle möglichen Sonderfälle, Grenzfälle betrachten)
- Testen und Verbessern
- Dokumentieren der Ergebnisse

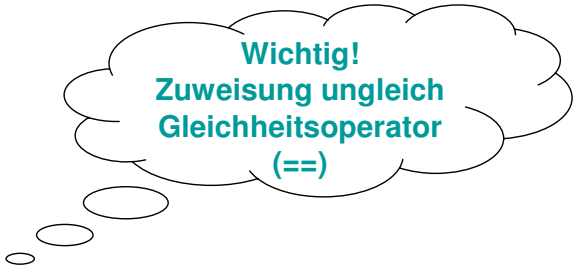
Java: Deklaration von Variablen

- In Java müssen alle Variablen vor der Verwendung deklariert werden
- Deklaration von Variablen
  - Angabe des Datentyps
  - Angabe des Namens (eindeutig)
  - [Angabe eine Anfangswertes (optional)]

```
int i;
int j = 0, k;
boolean positive = false;
float x;
char ch = ' ';
```

Java: Zuweisungsoperation

```
x = 1 + 2;
x = a + b;
x = x + b;
x = x + 1;
```



Weist einer Variable einen Wert zu.

- Der Wert der Variablen auf der rechten Seite geht dabei nicht verloren.
- Der Zuweisungsoperator ist nicht mit der Gleichheitsrelation der Mathematik zu verwechseln.
- Die Hauptanwendung des Zuweisungsoperators liegt in der Zuweisung eines Ergebnisses eines *Ausdruckes* an eine Variable.

Verschiedene Schreibweisen in unterschiedlichen PrgmSprachen

- `m = n` C/C++, **Java**, FORTRAN, Basic, ...
- `m <- n` APL
- `m := n` Pascal, Algol, Modula-2

Java: Sequenz

```
int number;
number = readInt();
number = number + 1;
writeInt(number);
...
```

Folge von Anweisungen werden nacheinander (sequentiell) ausgeführt

- Ist nichts anderes spezifiziert, werden Anweisungen immer nacheinander - man sagt ***sequentiell*** - ausgeführt
- Während der Ausführung werden Werte von Variablen verändert

Java: Verzweigungen

Eine Verzweigung führt Anweisungen in Abhängigkeit einer Bedingung aus.

Einseitig:

```
wenn x < 0 ist, dann
   x = -x
```

Bedingung

Anweisungen

Zweiseitig:

```
wenn x < y ist, dann
   min = x, max = y
sonst  /* x >= y */
   min = y, max = x
```

Java

```
if (x < 0) {
    x = -x
}
```

```
if (x < y) {
   min = x;  max = y;
} else { /* x >= y */
   min = y; max = x;
}
```

Java: Schleifen

Schleifen führen Anweisungen in Abhängigkeit einer Bedingung kein- oder mehrmals aus.

Beispiel: Berechnung der Fakultät

Abweiseschleife

Bedingung

```
x = 3, i = 1, f = 1
solange i < x ist, wiederhole
   i = i + 1, f = f * i
```

Anweisungen

Java

```
x = 3; i = 1; f = 1;
while (i < x) {
   i = i + 1; f = f * i;
}
```

Durchlaufschleife

```
x = 3, i = 1, f = 1;
wiederhole
   f  = f * i, i = i + 1
solange i <= x
```

```
x = 3; i = 1; f = 1;
do {
   f  = f * i; i = i + 1;
} while (i <= x);
```

Grundstruktur von Java-Programmen

```
class ProgramName {

  public static void main (String[] arg) {
    ... // Deklarationen
    ... // Anweisungen
  }

}
```

Text muß in einer Datei namens ***ProgramName*.java** stehen

main-Methode stellt den Anfang des Programms dar

Beispiel:

```
class Sample {
  public static void main (String[] arg) {
    Out.print("Geben Sie 2 Zahlen ein:");
    int a = In.readInt();
    int b = In.readInt();
    Out.println("Summe = " + (a + b));
  }
}
```

Text steht in Datei **Sample.java**

Algorithmenschreibweisen

+ vom Computer lesbar und verarbeitbar
+ eindeutig und präzise
+ von Menschen lesbar

- exakt, ohne Freiheitsgrade
- aufwendig zu schreiben

Java-Programm

```
public class GGTProgram {

  public static void main(String[] args) {

    int x;
    int y;
    int ggt;

    Out.print("Bitte x eingeben: ");
    x = In.readInt();
    Out.print("Bitte y eingeben: ");
    y = In.readInt();
    ggt = GGTEuklid(x, y);
    Out.print("Der GGT von x und y ist: ");
    Out.println(ggt);
  }

  static int GGTEuklid(int x, int y) {

    int rest;
    int ggt;

    // %-Operator bildet den Rest
    rest = x % y;
    while (rest != 0) {
      x = y;
      y = rest;
      rest = x % y;
    }

    // rest == 0
    ggt = y;
    return ggt;
  }
} // end GGTProgram
```

Java-Programm "Sum" (Sum.java)

```
public class Sum {

    public static void main(String[] args) {
        int sum = 0;
        int number;
        Out.print("Bitte Zahlen eingeben: ");
        number = In.readInt();
        while (number > 0) {
            sum = sum + number;
            number = In.readInt();
        }
        Out.print("Die Summe ist:");
        Out.printLn(sum);
    }

}
```

Einige Hinweise

- Es ist wichtig, klar und einfach zu denken
  - Hat man das Problem einmal richtig erfasst (niederschreiben), so ist meist auch die Lösung klar.
  - Die besten Lösungen sind (fast immer) die einfachsten und kürzesten.
- Sonderfälle und Grenzfälle betrachten (z.B. Division durch 0)
  - Fehler treten meist bei den Sonderfällen und Grenzfällen auf; an diese denken und im Algorithmus vorsehen.
  - Der Mensch berücksichtigt Sonderfälle meist intuitiv; Intuition fehlt dem Computer aber völlig.
- Java-Programme müssen absolut exakt, verständlich und übersichtlich sein
  - Keine Syntaxfehler
  - Klare Namensgebung
  - Genaue Konventionen für das Format und die Zeichensetzung (Einrückungen, Leerzeichen etc.)
  - Kommentare für Verständnis wichtig