INFORMATIK

Struktogramme = schrittweises Abarbeiten von Befehlen

Algorithmus = Schrittweises, präzises Verfahren zur Lösung eines Problems

Bausteine

Zuweisungen		$a \leftarrow 1$	Der Variablen a wird den Wert 1 zugewiesen.
Einlesen		Lies Zahl a	Liest die Zahl a ein.
Selektion		Bedingung / wahr / falsch / A1 / A2	Prüft ob eine Bedingung wahr oder falsch ist, und führt die dazugehörige Anweisung aus.
Mehrfachselektion		Variable / C1 C2 C3 Cn / A1 A2 A3 … An	Je nach Wert der Variable wird eine bestimmte Anweisung ausgeführt.
Repetition	vorangehend	Anweisung / Bedingung	Wird einmal durchgeführt. Danach wiederholt, bis Bedingung nicht mehr erfüllt wird.
	nachfolgend	Bedingung / Anweisung	Wird solange wiederholt, bis Bedingung nicht mehr erfüllt wird.

Vom Quellcode zum Prozess

MeinProg.java	→	MeinProg.class	→	o
(Quellcode)	(sofern keine Fehler)	Programm		java MeinProg

Schrittweises Verfeinern

Wie komme ich vom Problem zur Lösung?

Bsp. Gesucht sind alle vollkommenen (= Zahl, Summe ihrer echten Teiler) Zahlen bis 1000.

1. Jede Zahl zwischen 1 und 1000 prüfen, ob sie vollkommen ist.
   Wenn ja, ausgeben.
2. Schleife Zahl i von 1 bis 1000
   Summe der Teiler von i bestimmen
   Wenn gilt: Summe = i -> ausgeben

Zahlensysteme

10er-System	2er-System (Dualsystem; Binärsystem)	Hexadezimalsystem (16-er System)	
0,1,2,3,4,5,6,7,8,9,10	0,1	$0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F$	
231_{10}	$13_{10} \rightarrow 1101_2$	$165_{10} \rightarrow A5_{16}$	$1101'0100_2 \rightarrow D4_{16}$
$2*100+3*10+1*1$ $2*100^3+3*10^2+1*10^0$	$1*8+1*4+0*2+1*1$ $1*2^3+1*2^2+0*2^1+1*2^0$	$0*256+A*16+5*1$ $0*16^2+A*16^1+5*16^0$	$1101 \rightarrow D$ $0100 \rightarrow 4$

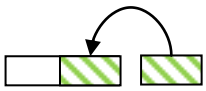
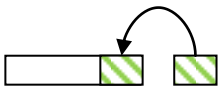
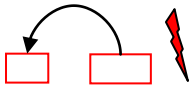
Beispiele

10er	2er	16-er
1	0001	1
2	0010	2
10	1010	A
16	0001'0000	10
20	0001'0100	14

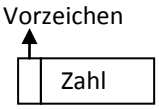
Ganze Zahlen in Java

Kompatibilität

$byte\ b;\ short\ s, int\ i;\ long\ l;$

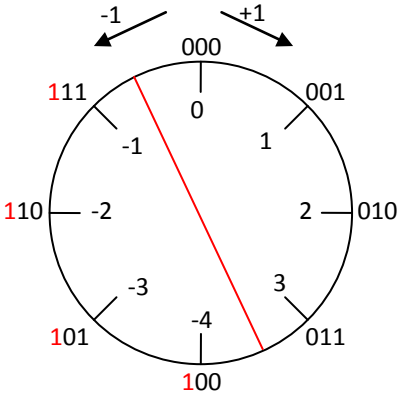
$i = s;$	$l = b;$	$b = s;$	Typenumwandlung $b = (byte)\ s;$	$s = s + 1$ $(+) = integer$
				

Andere Codierung

Vorzeichen/Betrags-Codierung	
	Problematik: - Doppelte Null - Arithmetik ist kompliziert $(++, --, -+, +-)$

Codierung in modernen Rechnern

Beispiel: 3 $Bits$



Eigenschaften

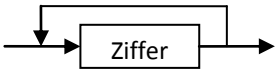
- n-Bits: 2^n Zahlen
- Vorderstes Bit = Vorzeichen
- eindeutige Null
- Wertebereich: -2^{n+1} … $+2^{n-1} - 1$
  Beispiel: $n = 8$: -128 … $+127$
- Arithmetik ohne Fallunterscheidung

$$\begin{array}{r} 3 \\ +\ -1 \\ \hline 2 \end{array}$$

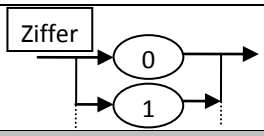
negative Zahlen:
Betrag als Dualzahl,
Invertieren (umkehren) der Bits
1 addieren

Syntaxnotationen

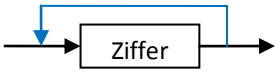
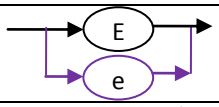
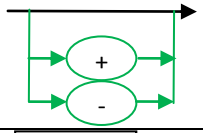
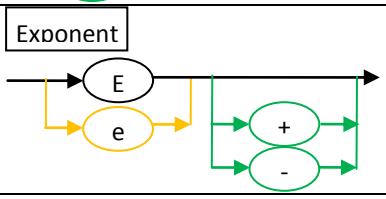
Möglichkeiten

1.	Zahl = Ziffer {Ziffer}	EBNF (Extended Backens Naur Form)
2.		Syntaxdiagramm

Symbole

Terminalsymbole sind Wörter einer Sprache. Sie werden nicht ersetzt.	Ziffer = „0“ | „1“ | „2“ | … | „9“
Nichtterminalsymbole werden ersetzt und dürfen im Text nicht vorkommen	

Regeln

	EBNF	Syntaxdiagramm
Wiederholungen	Zahl = {Ziffer}	
Alternative	(„E“ | „e“)	
Option	[„+“ | „-„]	
Beispiel	Exponent = („E“ | „e“) [„+“ | „-„] Digits Alternative Option	

Verzweigungen

EBNF

```
ifStatement = „if“ „(„ Expression „)“ Statement [„else“ Statement].
                      = Ausdruck        = wahr              = falsch
```

Bsp.

```
if (x>y) {
   max = x;
}else{
   max = y;
}
```

Boole’sche Ausdrücke

Wahrheitstabelle

Operanden		Und	Oder	Negation
x	y	x && y	x || y	!x
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

Kurzschlussauswertung

Sobald der Wert einer Berechnung fest steht, wird die Berechnung abgebrochen.

`Bsp. if (y!=0 && x/y>10) …`

`y!=0 -> Vorgehen bis zum zweiten und`

a) `y=0` -> Abbruch beim ersten Und, ansonsten wäre Division durch 0

`if (y==0 || 1/y<1) …`

a) y!=0 -> erste Bedingung ist falsch, Vorgang geht bis zum zweiten

b) y=0 –> erste Bedingung wahr, ganze Bedingung ist wahr

Hinweise

```
boolean p;
p = 0 < x < 10; →Error
p = 0 < x && x < 10; → Richtig
```

```
if (b == true= {…} → Kompliziert
if (b) {…} → Einfacher, 1 Operation weniger
```

Transformation

`while(i<10) { … }`	`if (i<10) { do{…}while(i<10)}`

Gleitkommazahlen

Teilmengen

$$\supset = Teilmenge\ von$$

$$double \supset float \supset long \supset int \supset short \supset byte$$

Abschneidungen

$f = i;$	$\rightarrow richtig$
$i = f;$	$\rightarrow falsch$
$i = (int)f;$	$\rightarrow gewollte\ Abschneidung$
$d = 3.14;$	$\rightarrow richtig$
$f = 3.14;$ $\{float = double\}$	$\rightarrow falsch$
$f = 3.14F;$ $f = (float)3.14;$	$\rightarrow richtig$

Standard

$1 \rightarrow int$

$1.0 \rightarrow double$

Methoden (Unterprogramm, Prozedur, Funktion)

lokale und globale Variablen (Namen)

Verdecken

```
class Prog2
{
   static int x;
   static int f(int x)
   {
      return x + 1;
   }

   p.s.v. main (String [] arg)
   {
      x = 5;
      int y = f(6);
   }
}
```

lokale Variable überwiegt

```
Ergebnis
x = 5
y = 7
```

Wann was?

local	standart + besser lesbar (Nähe), keine Namenskonflikte
global	Variable f mehrere Methoden (geteilt) Wert bleibt über mehrere Aufrufe erhalten

Überladen

```
static int max(int a, int b) {…}
static int max(int a, int b, int c) {…}
static double max(double a, double b) {…}
…
z = max(5,7);
z = max(5,7,3);
z = max(3.4, 7.5);
```