

Installation Java 1.4.2

erste Schritte

→ siehe Java Tutorial

CD / internet

Man geht alle Programme selber downloaden

## 2.3 Primzahl

### 2.3.1 Das Problem

Geg: Zahl  $n \in \mathbb{N}$   $n > 0$

Ges: kleinste Primzahl  $p$  mit  $p > n$

### 2.3.2 Algorithmus

Prüfe alle Zahlen  $z = n+1, n+2, \dots$  ob sie Primzahl sind. Die erste so gefundene Primzahl ist das Ergebnis.

Prüfe Zahlen  $k = 2, 3, 4, \dots$  (?) ob sie  $z$  ohne Rest teilen. Falls ja, so ist  $z$  keine Primzahl, falls nein für jedes solche  $k$ , so ist  $z$  eine Primzahl

○: Es reicht, mit  $k$  bis  $\lceil \sqrt{z} \rceil$  zu gehen

$\lceil \rceil \hat{=}$  aufrunden,  $\lfloor \rfloor$  abrunden

Beweis für ○:

Zeige: Wenn  $z$  einen Teiler  $\neq 1, z$  hat,  
dann hat  $z$  auch einen Teiler  $\leq \sqrt{z}$

Sei  $k$  Teiler von  $z$ ,  $k \neq 1, z$

$\Rightarrow$  kann  $z$  schreiben als  $z = k \cdot l$   $l \neq 1, z$   $l \in \mathbb{N}$

Annahme  $k > \sqrt{z}$  und  $l > \sqrt{z}$  } Widerspruch  
 $\Rightarrow k \cdot l > \sqrt{z} \cdot \sqrt{z} = z$  in  $k \cdot l = z$

$\Rightarrow k \leq \sqrt{z}$  oder  $l \leq \sqrt{z}$

"Beweis durch Widerspruch"

Algorithmus für Primzahltest der Zahl  $z$

Setze  $k := 2$

Setze teilerGefunden := falsch;  $\leftarrow$  boolesche Variable  
noch kein Teiler gefunden

Solange (teilerGefunden = falsch) und ( $k * k \leq z$ ) führe aus

teile  $z$  ganzzahlig durch  $k$ , sei rest der Rest

falls rest = 0, so setze teilerGefunden := wahr

setze  $k := k + 1$ ;

↗

"Pseudocode" ähnelt Pascal



und Verlassen der Schleife gilt

$z$  ist Primzahl  $\Leftrightarrow$  teilbarGefunden = falsch

Zahlenbeispiel

$$z = 17$$

$$k = 2 \quad \text{rest } 1$$

$$k = 3 \quad \text{rest } 2$$

$$k = 4 \quad \text{rest } 1$$

$$k = 5 \quad \text{rest } 2 \quad 5 \times 5 > 17$$

teilbarGefunden = falsch  $\Rightarrow z = 17$  Primzahl

$$z = 35$$

$$k = 2 \quad \text{rest } 1$$

$$k = 3 \quad \text{rest } 2$$

$$k = 4 \quad \text{rest } 3$$

$$k = 5 \quad \text{rest } 0 \quad \text{da } z = 5 \cdot 7$$

$\hookrightarrow$  teilbarGefunden = wahr  
abbruch der Schleife

$z$  keine Primzahl

äußere Schleife

Lese Zahl  $n$  ein

Setze  $z := n$

Wiederhole

Erhöhe z um 1

überprüfe ob z Primzahl ist (Pseudocode von oben)

bis z Primzahl ist

Gebe z aus

Erläuterung Java Code

siehe Coma WWW

Neu:

① Konstrukte für Schleifen

while ( Fortsetzungsbedingung ) Anweisung



Ausdruck, der  
boolean ergibt.  
Ist dieser "true" so  
wird Anweisung  
ausgeführt, danach  
Fortsetzungsbed.  
erneut überprüft.  
Ist dieser "false"  
so wird Anweisung  
nicht ausgeführt



beliebige Anweisung  
auch if, while --  
guter Stil Anweisung  
mit { ... }  
in Klammern

do Anweisung while (Fortsetzungsbedingung);

wird auf jeden Fall

1x ausgeführt

dann wird Fortsetzungsbed. überprüft

Falls "true", wird Anweisung

wiederholt usw.

② neue Operatoren:

! logische Negation

&& logisches und

% Rest bei ganzzahliger Division

TIPP: Überprüfung, dass gewünschte Programmablauf stattfindet  
bei komplizierten Schleifen und logischen Bedingungen

- Tabelle alle Variablen machen

- Schrittweise durch Programm gehen, Werte der <sup>entsp.</sup> Var. ändern  
und überprüfen ob sie noch die "richtigen" Werte  
anzeigen

n	z	k	divisorfound	Operatoren
17	-	-	-	einlesen von n
17	17	-	-	z = n gesetzt
17	18			z++
17	18	2		k = 2

17 18 2 false

divisorFound = false

← Einheitsschleife da

$(\underbrace{! \text{divisorFound}}_{\text{true}}) \&\& (\underbrace{k*k \leq z}_{2*2 \leq 18})$   
true



"Buchhaltung", unterstützt durch Debugger



Später in Eclipse

im Skript vollständig für Primzahl.java