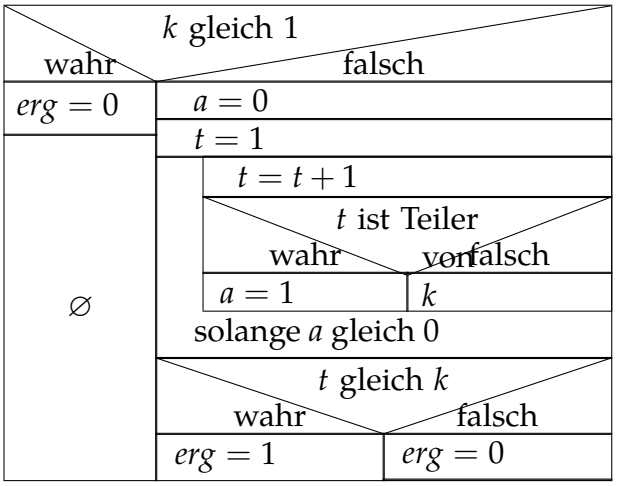
Abitur 2017 IV

(*Primzahl*)

Stichwörter: Ein-Adress-Befehl-Assembler

In dem folgenden Struktogramm wird ein Algorithmus dargestellt, der erkennt, ob eine natürliche Zahl k eine Primzahl ist. In diesem Fall wir in die Speicherzelle *erg* die Zahl 1 abgelegt, sonst 0.



(a) Stellen Sie die Veränderung der Variablenwerte bei Ablauf dieses Algorithmus jeweils für die Startwerte $k = 5$ und $k = 15$ durch zwei Speicherbelegungstabellen wie nachfolgend gezeigt dar.

Anweisung	k	a	t	erg
	5			
$a = 0$		0		
$t = 1$			1	
$t = t + 1$	5		2	

Lösungsvorschlag

$k = 5$

Anweisung	k	a	t	erg
	5			
$a = 0$		0		
$t = 1$			1	
$t = t + 1$			2	
$t = t + 1$			3	
$t = t + 1$			4	
$t = t + 1$			5	
$a = 1$		1		
$erg = 1$				1

$k = 15$

Anweisung	k	a	t	erg
	15			
$a = 0$		0		
$t = 1$			1	
$t = t + 1$			2	
$t = t + 1$			3	
$t = t + 1$			4	
$t = t + 1$		1	5	
$a = 1$		1		
$erg = 0$				0

Im Folgenden soll ein Programm für diese Maschine erstellt werden, das den dargestellten Algorithmus umsetzt. Der Wert von k soll in Speicherzelle 101, der von a in 102, der von t in 103 und der von erg in 104 gespeichert werden.

(b) Betrachten Sie die folgende kurze Sequenz; xx steht dabei für ein geeignetes Sprungziel.

```
LOAD 101
MOD 103
JMPP xx
LOADI 1
STORE 102
```

Geben Sie an, welcher Teil des Algorithmus damit umgesetzt wird.

Lösungsvorschlag

Die Bedingung „t ist Teiler von k“.

(c) Setzen Sie unter Verwendung der Sequenz aus Teilaufgabe 2b den gesamten Algorithmus in eine Programm für die gegebene Registermaschine um.

Lösungsvorschlag

Assembler

```
# k: 101
# a: 102
# t: 103
# erg: 104

# IF k = 1 THEN
                LOADI 101
                CMPI 1
                JMPZ ist_nicht_prim

# a := 1;
                LOADI 0
                STORE 102

# t := 1;
                LOADI 1
                STORE 103

# t := t + 1;
erhoehe_t:      LOAD 103
                ADDI 1
                STORE 103

# IF (k % t) = 0 THEN
                LOAD 101
                MOD 103
                JMPP solange_bed

# a := 1;
                LOADI 1
                STORE 102

# UNTIL a = 0;
solange_bed:    LOAD 102
```

```
                JMPZ erhoehe_t

# IF t = k THEN;
                LOAD 103
                CMP 101
                JMPZ ist_prim

# erg := 0;
ist_nicht_prim: LOADI 0
                STORE 104
                JMP ende

# erg := 1;
ist_prim:       LOADI 1
                STORE 104

ende:           HOLD
```

Minisprache

```
PROGRAM primzahl;
VAR k, a, t, erg;

BEGIN
  k := 5;
  IF k = 1 THEN
    erg := 0;
  ELSE
    a := 0;
    t := 1;
    REPEAT
      t := t + 1;
      IF (k % t) = 0 THEN
        a := 1;
      END
    UNTIL a = 0;

    IF t = k THEN
      erg := 1;
    ELSE
      erg := 0;
    END
  END;
END primzahl.
```

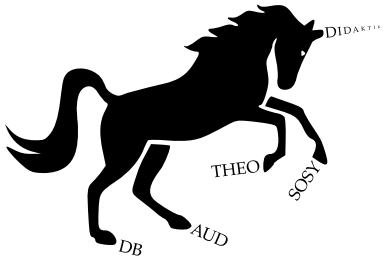
Java

```
  public static boolean istPrimzahl(int k) {
    if (k == 1)
      return false;
    int a = 0;
    int t = 1;
```

```
    do {
      t++;
      if (k % t == 0) {
        a = 1;
      }
    } while (a == 0);

    if (t == k) {
      return true;
    } else {
      return false;
    }
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/aufgaben/tech_info/assembler/ein_adress/Primzahl.java`



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.Der TEX-Quelltext dieses Dokuments kann unter folgender URL aufgerufen werden: `https://github.com/bschlangaul-sammlung/examens-aufgaben/blob/main/Module/50_TECH/10_Ein-Adress/Aufgabe_05-Abitur-2017-IV-Primzahl.tex`