3.3 Testen und Verbessern von Programmen

Primzahlsuche 2

Das letzte Kapitel endete mit dem Satz "Diese Lösung ist wohl leicht verständlich, aber doch noch wesentlich verbesserungsfähig." Daran soll jetzt angeknüpft werden.

So ist leicht einsehbar, dass nicht alle Zahlen bis n-1 auf Teilereigenschaft überprüft werden müssen, sondern nur bis n/2. Mit der folgenden Überlegung lässt sich zeigen, dass es schon genügt, die Zahlen t bis zu $\sqrt{n}$ auf Teilereigenschaft bzgl. n zu untersuchen.

Wäre nämlich eine Zahl t mit $t > \sqrt{n}$ ein Teiler von n, dann gälte $\frac{n}{t} \cdot t = n$; es wäre dann auch $\frac{n}{t}$ein Teiler von n, wobei $\frac{n}{t}$ kleiner als $\sqrt{n}$ wäre. Alle Zahlen $< \sqrt{n}$ sind aber in diesem Stadium bereits überprüft, und der Teiler $\frac{n}{t}$ wäre schon gefunden!

In einer ersten Verbesserung des Programms muss daher lediglich die Abbruchbedingung korrigiert werden; die Rechenzeit kann dadurch merklich verkürzt werden.

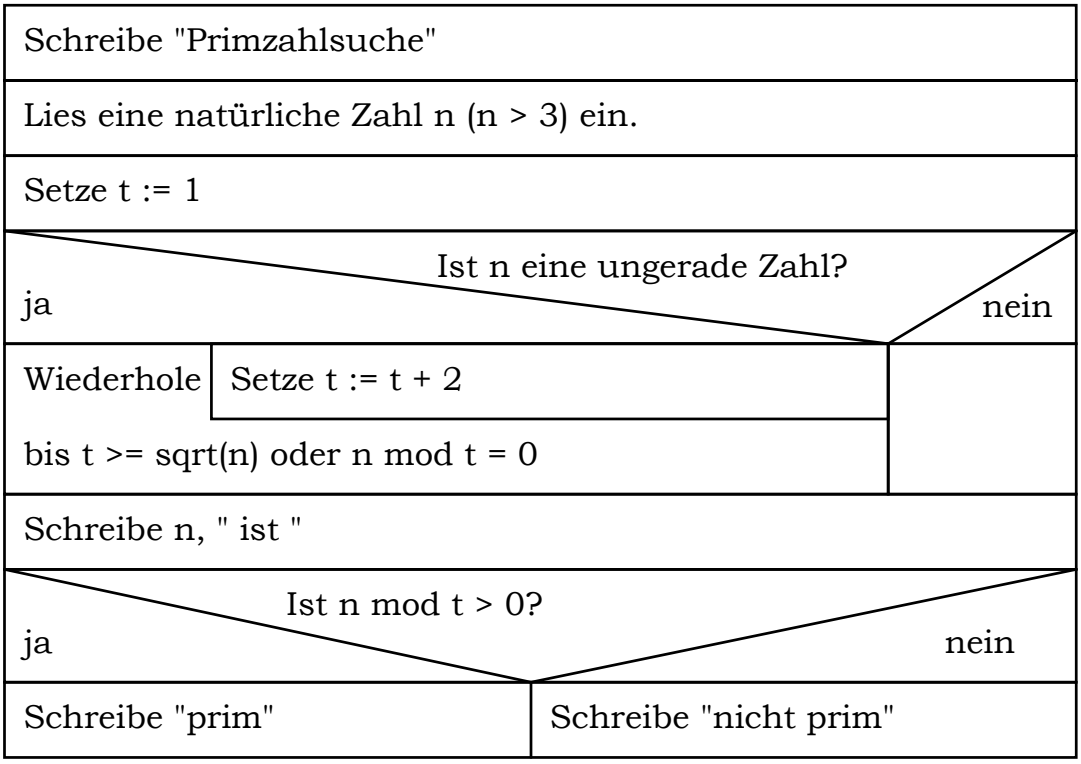
Die entsprechende Programmzeile lautet dann:

```
... until (t = trunc(sqrt(n))) or (n mod t = 0); ...
```

Anmerkung:
Die Wurzel aus n ist nicht notwendigerweise eine natürliche Zahl, wie von t verlangt; Funktion **TRUNC** wandelt durch Abschneiden der Nachkommastellen eine reelle Zahl in eine Zahl vom Longint um.
Syntax:
TRUNC*(Realvariable)*;

Eine weitere Verkürzung der Rechendauer bringt die Auswertung der Überlegung, dass mit Ausnahme der Zahl 2 keine weiteren geraden Zahlen auf Teilereigenschaft geprüft werden müssen. Wäre nämlich eine natürliche Zahl z. B. durch 18 teilbar, dann natürlich auch durch 2; die Zahl wäre dann bereits als mögliche Primzahl ausgeschieden. Es genügt daher, die Teilbarkeit durch 2 und ungerade Zahlen zu untersuchen. Am Anfang der Teilbarkeitsprüfung wird deshalb geprüft, ob die zu untersuchende Zahl gerade ist; falls nicht, läuft der bekannte Prüfalgorithmus (nur mehr mit ungeraden Zahlen!) ab. Es handelt sich in diesem Beispiel also um eine teilweise geschachtelte Auswahl.

So sieht das zugehörige Struktogramm aus:



Das Pascal-Programm:

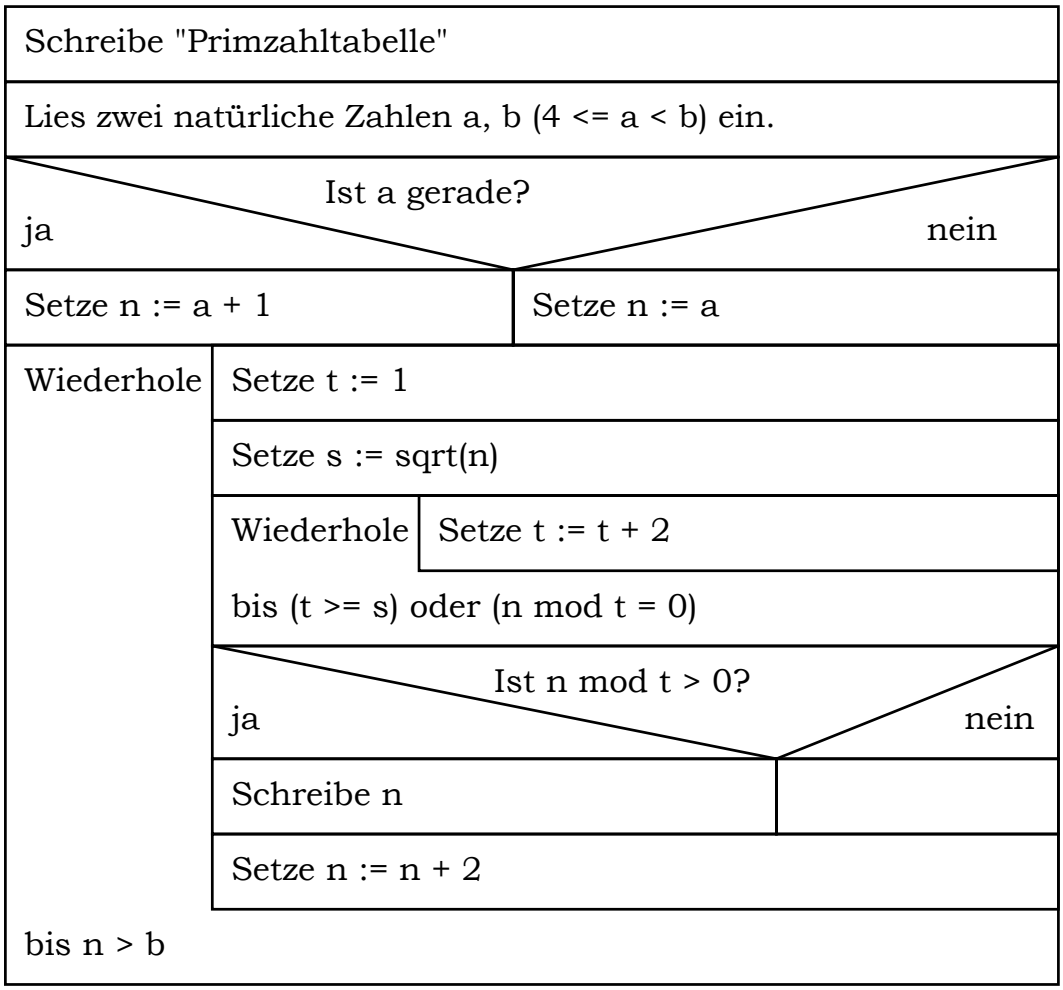
```
program programm13_(primzahlpruefung_2);
uses crt;
var  n, t:longint;
     antwort:  char;
begin
     repeat
          clrscr;
          writeln ('Primzahlprüfung 2');
          writeln;
          write('Gib eine natürliche Zahl (n > 3) ein. ');
          readln(n);
          t := 1;
          if n mod 2 > 0 then
          repeat
               t := t + 2
          until (t >= trunc(sqrt(n))+1) or (n mod t = 0);
          write (n, 'ist ');
          if n mod t > 0 then writeln ('prim.')
                         else writeln ('nicht prim.');
          writeln;
          writeln ('Nochmals? (j/sonst. Taste)');
          antwort := readkey
     until antwort <> 'j'
end.
```

Anmerkung zur Programmierung:
Zunächst hat die Variable t (der mögliche Teiler) den Wert 1. Gilt n mod 2 = 0, dann wird die weitere Prüfung übersprungen und wegen t = 1 "nicht prim" ausgegeben; andernfalls wird die Prüfung (mit ungeraden Teilern) fortgesetzt.

Primzahlsuche 3

Wegen der großen Ausführungsgeschwindigkeit der Primzahlprüfung fallen die zuletzt genannten Verbesserungen kaum auf. Dies ändert sich deutlich, wenn viele Prüfungen durchzuführen sind, zum Beispiel bei der Ermittlung der Primzahlen in einem Intervall. Dann müssen nämlich alle Zahlen im Intervall auf Primzahleigenschaft geprüft werden.

Graphische Darstellung (Struktogramm)



Um einen Seitenumbruch innerhalb der Darstellung des Struktogramms zu vermeiden, werden die im Sinne der strukturierten Programmierung anzustellenden neuen Überlegungen hier angeschrieben:

1. Festlegung der Objekte:
Es sollen nur natürliche Zahlen >= 3 eingegeben werden, um den obigen Algorithmus im wesentlichen beibehalten zu können. Ausgegeben sollen die Primzahlen aus dem Intervall werden.

2. Grobformulierungen der Problemlösung:
In einer Schleife werden die ungeraden Zahlen mit ungeraden Teilern geprüft.

Das Pascal-Programm:

```
program programm14_(primzahltabelle);
uses crt;
var  n, t, s, a, b: longint;
     antwort:       char;
begin
     repeat
          clrscr;
          writeln('Primzahltabelle');
          writeln;
          write('von   bis? ');
          readln (a, b);
          if a mod 2 = 0 then n := a+1
                         else n := a;
          repeat
               t := 1;
               s := trunc(sqrt(n))+1;
               repeat
                    t := t + 2
               until (t >= s) or (n mod t = 0);
               if n mod t > 0 then write (n:8);
               n := n+2
          until n > b;
          writeln;
          writeln ('Nochmals? (j/sonst. Taste)');
          antwort := readkey
     until antwort <> 'j'
end.
```