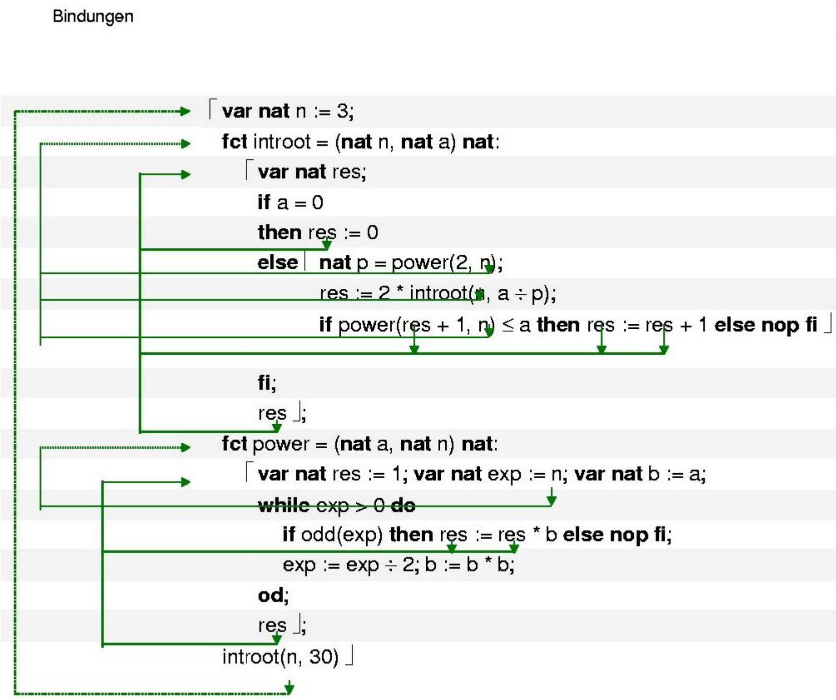
Übungen zu Einführung in die Informatik I

Aufgabe 53 Bindungen, Lebensdauer, Gültigkeit (Lösungsvorschlag)

Die jeweiligen Bindungen sind durch Pfeile dargestellt, die für die verschiedenen Bezeichner unterschiedliche Linien haben. Lebensdauer und Bindungen sind durch die Punkte in der Tabelle dargestellt.



Bindungen	Lebensdauer							G ltigkeit						
	n	n	n	res	res	p	power	n	n	n	res	res	p	power
⌈ **var nat** n := 3;	•						•	•						•
fct introot = (**nat** n, **nat** a) **nat**:	•	•					•		•					•
⌈ **var nat** res;	•	•		•			•		•		•			•
if a = 0	•	•		•			•		•		•			•
then res := 0	•	•		•			•		•		•			•
else ⌈ **nat** p = power(2, n);	•	•		•		•	•		•		•		•	•
res := 2 * introot(n, a ÷ p);	•	•		•		•	•		•		•		•	•
if power(res + 1, n) ≤ a **then** res := res + 1 **else nop fi** ⌋	•	•		•		•	•		•		•		•	•
fi;	•	•		•			•		•		•			•
res ⌋;	•	•		•			•		•		•			•
fct power = (**nat** a, **nat** n) **nat**:	•		•				•			•				•
⌈ **var nat** res := 1; **var nat** exp := n; **var nat** b := a;	•		•		•		•			•		•		•
while exp > 0 **do**	•		•		•		•			•		•		•
if odd(exp) **then** res := res * b **else nop fi**;	•		•		•		•			•		•		•
exp := exp ÷ 2; b := b * b;	•		•		•		•			•		•		•
od;	•		•		•		•			•		•		•
res ⌋;	•		•		•		•			•		•		•
introot(n, 30) ⌋	•						•	•						•

Aufgabe 54 Bedingte Anweisung und Wiederholungsanweisung (Lösungsvorschlag)

Wir verwenden einen sehr einfachen Algorithmus, der mit einer *gezählten Wiederholungsanweisung* für alle Zahlen $1 \leq i \leq n$ testet, ob i prim ist und i mit Hilfe einer *bedingten Anweisung* auf dem Bildschirm ausgibt, falls der Test ein positives Ergebnis liefert. Für den Primzahltest formulieren wir eine *Prozedur*. Wir entwickeln zunächst das Hauptprogramm und anschließend die Prozedur für den Primzahltest.

```
PROGRAM prim (Input, Output);

VAR max,i : INTEGER;
    test  : BOOLEAN;

PROCEDURE istPrim (n : INTEGER; VAR erg: BOOLEAN);
VAR i : INTEGER;
```

```
BEGIN
   IF (n < 2)                        { falls n < 2 }
      THEN erg := FALSE              { dann ist n keine Primzahl }
   ELSE BEGIN
         i := 2;
         erg := TRUE;                { bisher Prim-Eigenschaft nicht widerlegt }
         WHILE (i <= n/2) AND erg    { testen bis widerlegt oder i >= n/2      }
            DO BEGIN
               IF n MOD i = 0
                  THEN erg := FALSE;
               i := i+1
            END
      END
END;

BEGIN
  Write("Obere Grenze? "); ReadLn(max);
  FOR i := 1 TO max
     DO BEGIN
        istPrim(i,test);
        IF test THEN
        BEGIN
           Write(i);
           WriteLn;
        END
     END
END.
```

Aufgabe 55 Der Softwareentwicklungsprozess (Lösungsvorschlag)

a) Der größte gemeinsame Teiler zweier natürlicher Zahlen a und b, mit $a > 0$ und $b > 0$, ist folgendermaßen spezifiziert:

fct ggT = (**nat** a, **nat** b: a > 0 ∧ b > 0) **nat**:
  some nat g: P(g, a, b)

wobei das Prädikat P durch

$$P(g,a,b) = g > 0 \land a \,\mathtt{mod}\, g = 0 \land b \,\mathtt{mod}\, g = 0 \\ \land \forall \,\mathbf{nat}\, z : [(a \,\mathtt{mod}\, z = 0 \land b \,\mathtt{mod}\, z = 0) \Rightarrow g \,\mathtt{mod}\, z = 0]. \tag{1}$$

gegeben ist.

b) Wie bereits aus der Vorlesung und den Übungen bekannt, gilt für den $\mathtt{ggT}(a,b)$:

$$\mathtt{ggT}(a,b) = \begin{cases} a & \text{für } a = b; \\ \mathtt{ggT}(a-b,b) & \text{für } a > b; \\ \mathtt{ggT}(a,b-a) & \text{für } b > a; \end{cases} \tag{2}$$

Die drei in der Rechenvorschrift (2) enthaltenen Identitäten sind nun zu zeigen:

(i) **a = b**: Einsetzen von $g = a$ in das Prädikat (1) ergibt:

$$a > 0 \land a \,\mathtt{mod}\, a = 0 \land b \,\mathtt{mod}\, a = 0 \\ \land \forall \,\mathbf{nat}\, z : [(a \,\mathtt{mod}\, z = 0 \land b \,\mathtt{mod}\, z = 0) \rightarrow a \,\mathtt{mod}\, z = 0].$$

Da laut Voraussetzung $a > 0$ ist, ist dieses Prädikat erfüllt.

(ii) **a > b**: Zu zeigen ist, dass $P(g,a,b) \Leftrightarrow P(g,a-b,b)$. Wir betrachten das Prädikat (1):

$$\begin{aligned} P(g,a,b) = g > 0 \,\wedge\, a \,\mathtt{mod}\, g = 0 \,\wedge\, b \,\mathtt{mod}\, g = 0 \\ \wedge\, \forall\, \mathbf{nat}\, z : [(a \,\mathtt{mod}\, z = 0 \,\wedge\, b \,\mathtt{mod}\, z = 0) \rightarrow g \,\mathtt{mod}\, z = 0]. \end{aligned} \tag{3}$$

Allgemein gilt für natürliche Zahlen a,b und x:

$$\begin{aligned} & a \,\mathtt{mod}\, x = 0 \,\wedge\, b \,\mathtt{mod}\, x = 0 \\ \Leftrightarrow\quad & a \,\mathtt{mod}\, x - b \,\mathtt{mod}\, x = 0 \,\wedge\, b \,\mathtt{mod}\, x = 0 \\ \Leftrightarrow\quad & (a-b) \,\mathtt{mod}\, x = 0 \,\wedge\, b \,\mathtt{mod}\, x = 0 \end{aligned}$$

Wenden wir dies auf das Prädikat (3) an, dann folgt:

$$\begin{aligned} g > 0 \,\wedge\, (a-b) \,\mathtt{mod}\, g = 0 \,\wedge\, b \,\mathtt{mod}\, g = 0 \\ \wedge\, \forall\, \mathbf{nat}\, z : [((a-b) \,\mathtt{mod}\, z = 0 \,\wedge\, b \,\mathtt{mod}\, z = 0) \rightarrow g \,\mathtt{mod}\, z = 0]; \end{aligned} \tag{4}$$

Bei Prädikat (4) handelt es sich jedoch um $P(g,a-b,b)$ und die Äquivalenz $P(g,a,b) \Leftrightarrow P(g,a-b,b)$ für $a > b$ ist gezeigt.

(iii) **a < b**: Der Beweis erfolgt analog wie im Fall **a > b**.

c) **fct** ggT = (**nat** a, **nat** b: a>0, b>0) **nat**:

```
⌈   var nat j,k,result;
    j := a;
    k := b;

    while j ≠ k do
        if j > k then j := j - k
               else k := k - j
        fi
    od;
    result := j;
    result                          ⌋
```

d) **Lösungsstrategie:**

- Identifizieren der Invariante; diese ist bereits direkt vor der `while` – Schleife angegeben.
- Annotation ans Ende der `while` – Schleife setzen ($I \wedge \neg B$).
- Annotationen an den Beginn und ans Ende des Schleifenkörpers setzen ($I \wedge B$ und I, wobei `B` die Schleifenbedingung ist).
- Annotieren der bedingten Anweisung; d.h zeigen, dass die Formeln $\{I \wedge B \wedge (j > k)\} j = j - k \{I\}$ und $\{I \wedge B \wedge \neg(j > k)\} k = k - j \{I\}$ gelten.

fct ggT = (**nat** a, **nat** b: a>0, b>0) **nat**:

```
⌈   var nat j,k,result;
                        { a > 0 ∧ b > 0 }
                        { a > 0 ∧ b > 0 ∧ ggT(a, b) = ggT(a,b) }
    j := a;
                        { j > 0 ∧ b > 0 ∧ ggT(a, b) = ggT(j,b) }
    k := b;
```

$\{ j > 0 \wedge k > 0 \wedge ggT(a, b) = ggT(j,k) \}$

while $j \neq k$ **do**

$\{ j > 0 \wedge k > 0 \wedge ggT(a, b) = ggT(j,k) \wedge j \neq k \}$

if $j > k$ **then**

$\{ j > 0 \wedge k > 0 \wedge ggT(a, b) = ggT(j,k) \wedge j > k \}$

$\{ k > 0 \wedge ggT(a, b) = ggT(j-k,k) \wedge (j-k)>0 \}$

$j := j - k$

$\{ k > 0 \wedge ggT(a, b) = ggT(j,k) \wedge j > 0 \}$

else

$\{ j > 0 \wedge k > 0 \wedge ggT(a, b) = ggT(j,k) \wedge j < k \}$

$\{ j > 0 \wedge ggT(a, b) = ggT(j,k-j) \wedge 0<(k-j) \}$

$k := k - j$

$\{ j > 0 \wedge ggT(a, b) = ggT(j,k) \wedge 0 < k \}$

fi

$\{ j > 0 \wedge k > 0 \wedge ggT(a, b) = ggT(j,k) \}$

od;

$\{ j > 0 \wedge k > 0 \wedge ggT(a, b) = ggT(j,k) \wedge (j = k) \}$

$\{ j > 0 \wedge ggT(a, b) = ggT(j, j) \}$

$\{ j > 0 \wedge ggT(a, b) = j \}$

result := j;

$\{ result > 0 \wedge result = ggT(a, b) \}$

result $\rfloor$

e)
```
function ggT (a : Integer; b : Integer) : Integer;

var j, k, result : Integer;

begin
   j := a;
   k := b;

   while not(j = k) do
   begin
      if j > k then j := j - k
      else k := k - j
   end;

   result := j;
   ggT := result

end; { ggT }
```

Folievorlage für Aufgabe 55

```
fct ggT = (nat a, nat b: a>0, b>0) nat:
  ⌈  var nat j,k,result;
                          { a > 0 ∧ b > 0 }
                          { a > 0 ∧ b > 0 ∧ ggT(a, b) = ggT(a,b) }
     j := a;

     k := b;
                          { j > 0 ∧ k > 0 ∧ ggT(a, b) = ggT(j,k) }

     while j ≠ k do

          if j > k then

               j := j - k

               else

               k := k - j

          fi

     od;

     result := j;
                          { result > 0 ∧ result = ggT(a, b) }
     result                                                    ⌋
```