 3. März 2004

Aufgabe 4.13

Dem Lehrbuch entsprechend sind nachfolgend die Anfangswerte von Variablen durch Unterstreichung gekennzeichnet. Für $x_1 = \underline{x}_1$ heißt das z.B, dass die Variable x_1 bei der Initialisierung bzw. Übergabe den Wert $\underline{x}_1$ hat.

Block *Sortieren* (**mit:** $\{x_i \in \mathbf{Z}\}_{i = 1 \ldots N}$)
vor $\{x_1 = \underline{x}_1, x_2 = \underline{x}_2, \ldots x_{N-1} = \underline{x}_N : \underline{x}_i \in \mathbf{Z}\}$
nach $\{x_i = \underline{x}_m, x_{i+1} = \underline{x}_n : \bigwedge_{i-1} \underline{x}_m \leq \underline{x}_n ; \underline{x}_m, \underline{x}_n \in \{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_N\}\}$

Aufgabe 4.14 (⇒ Progammdatei: A4_14C.CPP)

a) Sei z.B. $x = 5$. Dann wird die Nachbedingung $y^2 = x$ durch $y = \pm\sqrt{5} = \pm 2{,}236$ erfüllt. Dies ist jedoch keine ganze Zahl. Für $x = 5$ ist zwar die Vorbedingung $x > 0$ erfüllt, aber die Nachbedingung $y^2 = x$ ist für ganzzahlige y nicht erfüllbar.

b) Eine korrekte Spezifikation erhält man durch eine strengere Vorbedingung:

Block *root* (**ein:** $x \in \mathbf{N}$; **aus:** $y \in \mathbf{N}$)
vor $x = \underline{x}^2, \underline{x} \in \mathbf{N}$
nach $y = \underline{x}$;

c) Nach Beispiel 4.9 gilt: $n^2 = \sum_{i=1,3,5,\ldots}^{2n-1} i$.

Es ist z.B. $5^2 = 1 + 3 + 5 + 7 + 9 = 25$.

Umgekehrt erhält man die ganzzahlige Quadratwurzel aus 25 nach dem folgenden Schema

25 - 1 = 24
24 - 3 = 21
21 - 5 = 16
16 - 7 = 9
9 - 9 = 0 ⇒ $\sqrt{25} = (9 + 1) / 2 = 5$.

Dieses Berechnungsschema wird durch den folgenden Block verallgemeinert:

```
Block root (ein: x ∈ N; aus: y ∈ N)
        Daten i ∈ N;
{       i := 1;
        Solange x ≠ 0 führe aus
        {   x := x - i;
            i := i + 2;
        }
        y := (i - 1) / 2;
}
```

Man beachte, daß in der letzten Zuweisung *(i - 1)* statt *(i + 1)* stehen muß, da im letzten Durchlauf der Schleife *i* nochmals um 2 erhöht wurde.

Aufgabe 4.15

a) Ist eine Zusicherung (Aussage über den Wert von *a*).
b) Ist keine Zusicherung. Es wird keine Aussage über die Belegung von *b*, sondern eine Aussage über die nächste Aktion gemacht.
c) Ist eine Zusicherung (die Belegung von a hat sich nicht geändert).
d) Ist eine Zusicherung (Aussage über den Wert von *a*).
e) Ist keine Zusicherung (keine Aussage über Variablen, nicht präzise)

Aufgabe 4.16

a)
```
!! Pa: a > 0, b = 0;
Solange a ≠ b führe aus
{       a := a + b;
        b := b + a;
}
```

b)
```
!! Pb: a > 0, b > 0; a ≠ b;
Solange a ≠ b führe aus
{       a := a + b;
        b := b + a;
}
```

Dann gilt

!! Qa: $a = b, a, b > 0;$

!! Qb: Es kann keine sinnvolle Zusicherung angegeben werden, da die Schleife nicht terminiert.

Aufgabe 4.17

Lösung folgt später.

Aufgabe 4.18

a)
```
Block fib_iter ( ein: n ∈ N, aus: fib ∈ N );
    Daten k,                                  // Funktionsargument
         f,                                   // f = fib(k)
         g,                                   // g = fib(k-1)
         h ∈ N;                               // Hilfsgröße
    {
    Falls n = 0
    dann fib := 0                             !! fib = 0 = fib(0)
    sonst {  k = 1;
             f = 1;                           !! f = 1 = fib(1) = fib(k)
             g = 0;                           !! g = 0 = fib(0) = fib(k-1)
             Solange k ≠ n führe aus
             {       k := k + 1;              !! f = fib(k-1), g = fib(k-2)
                     h := f + g;              !! h = fib(k-1) + fib(k-2) = fib(k)
                     g := f;                  !! g = fib(k-1)
                     f := h;                  !! f = fib(k)
             }
         }
         fib := f;                            !! fib = fib(k) ∧ k = n

    }.
```

b) $I(f, g, k) \equiv (f = \text{fib}(k), g = \text{fib}(k\text{-}1))$ ist Schleifeninvariante, denn

(1) $P(f, g, k) \equiv (k = 1, f = 1, g = 0) \Rightarrow g = 0 = \text{fib}(0) = \text{fib}(k\text{-}1) \wedge f = 1 = \text{fib}(1) = \text{fib}(k)$
$\equiv (f = \text{fib}(1), g = \text{fib}(0), k{=}1) \equiv I(f, g, k = 1).$

(2) Der Schleifenkörper bewirkt die Datentransformation (siehe Zusicherungen in a))

A(f, g, k) = [f = fib(k), g = fib(k-1)]. Daher gilt

I(A(f, g, k)) ≡ (f = fib(k), g = fib(k-1) ∧ ($k < n$)

(3) Die Eignung der Invarianten zeigt c)

c) Nach den bisherigen Ausführungen gilt am Ende eines jeden Schleifendurchlaufs f = fib(k). Da k durch die Zuweisung *k := k + 1;* von eins an monoton steigt, gilt wegen der Abbruchbedingung $k \neq n$ nach Beendigung der Schleife $k = n$. Nach Ausführung der Zuweisung *fib := f;* gilt somit *fib* = f = fib(k). Diese Beziehung gilt wegen der anfangs gemachten Fallunterscheidung auch für $n = 0$.

d) Die Hilfsvariable h kann entfallen, wenn der Schleifenkörper wie folgt gestaltet wird:

```
{     k := k + 1;          !! f = fib(k-1), g = fib(k-2)
      fib := f + g;        !! fib = fib(k-1) + fib(k-2) = fib(k)
      g := f;              !! g = fib(k-1)
      f := fib;            !! f = fib(k)
}
```

Aufgabe 4.19 (⇒ Progammdatei: A4_19B.CPP)

a) Berechnung von $y = x^{2^n}$

Spezifikation: **Block** *pot_dual (* **ein**: $x \in \mathbf{R}$, $n \in \mathbf{N}_0$; **aus**: $y \in \mathbf{R}$ *);*
nach $y = x^{2^n}$;

Zahlenbeispiel:
$i = 0 \Rightarrow x^{2^0} = x$
$i = 1 \Rightarrow x^{2^1} = x \cdot x = (x^{2^0})^2$
$i = 2 \Rightarrow x^{2^2} = (x \cdot x) \cdot (x \cdot x) = (x^{2^1})^2$
$i = 3 \Rightarrow x^{2^3} = \left[(x \cdot x) \cdot (x \cdot x)\right] \cdot \left[(x \cdot x) \cdot (x \cdot x)\right] = (x^{2^2})^2$

Das Zahlenbeispiel legt eine iterative Lösung nahe.

1. Entwurf:

```
Block pot_dual ( ein: x ∈ R, n ∈ N0; aus: y ∈ R );
      Daten i ∈ N0;
{     Falls n > 0
      dann {  ?? Schleifeninitialisierung ??
              !! Anfangszustand x = x_unter^(2^0)
              Solange i < n führe aus
              {  ?? Erhöhung von i, Behandlung von x
                 !! x = x_unter^(2^i)
              }
           }
      y := x;
}
```

Algorithmus: **Block** *pot_dual (* **ein**: $x \in \mathbf{R}$, $n \in \mathbf{N}_0$; **aus**: $y \in \mathbf{R}$ *);*

```
        Daten i ∈ N0;
{       Falls n > 0
        dann {  i := 0;                          !! Anfangszustand x = x^(2^0)
                Solange i < n führe aus
                {  i := i + 1;
                   x := x * x;
                }
             }
        y := x;
}
```

Verifikation:

Dann gilt wegen der Initialisierung $i := 0$ und wegen $x = \underline{x}^{2^0} = \underline{x} = \left[\underline{x}^{2^{0-1}}\right]^2 = \left[\underline{x}^{1/2}\right]^2$ die Invariante vor dem ersten Durchlauf der Schleife.

Im Schleifenkörper wird zunächst i um Eins erhöht und dann der Inhalt von x quadriert:
$A(i, x) = [i+1; x \cdot x]$.

Somit gilt für den Inhalt x_1 von x am Ende des 1.ten Schleifendurchlaufs:

$$x_1 = \underline{x} \cdot \underline{x} = \underline{x}^{2^0} \cdot \underline{x}^{2^0} = \left[\underline{x}^{2^0}\right]^2 = \underline{x}^{2^1}; \qquad i = 1.$$

Annahme: Am Ende des i.ten (i < n) Durchlaufs gelte $x_i = \underline{x}^{2^i}$, $i < n$-1.
Dann folgt

$$x_{i+1} = x_i \cdot x_i = \underline{x}^{2^i} \cdot \underline{x}^{2^i} = \left[\underline{x}^{2^i}\right]^2 = \underline{x}^{2^{i+1}}; \quad i < n.$$

Nach Beendigung der Schleife ($i = n$) gilt schließlich:

$$x_n = \left[\underline{x}^{2n-1}\right]^2 = \underline{x}^{2^n}.$$

b) Die algorithmische Grundidee beruht auf der Dualzahldarstellung des Exponenten n und wird anschaulich an einem Zahlenbeispiel illustriert:

$$y(x, 35) = x^{35} = x^{32} \cdot x^2 \cdot x^1 \cdot = ((((x^2)^2)^2)^2)^2 \cdot x^2 \cdot x^1 = x^{2^5} \cdot x^{2^1} \cdot x^{2^0} \quad (35_{10} = 100011_2)$$

Zur Ermittlung der Dualzahldarstellung bietet sich das Verfahren der fortgesetzten ganzzahligen Division durch 2 mit Rest an:

```
35 = 2 · 17 + 1
17 = 2 ·  8 + 1
 8 = 2 ·  4 + 0
 4 = 2 ·  2 + 0
 2 = 2 ·  1 + 0
 1 = 2 ·  0 + 1      ⇒      35₁₀ = 100011₂ (= z5 z4 ... z0)
```

Immer wenn bei der ganzzahligen Division durch 2 ein Rest entsteht, liegt eine 2er-Potenz (Dualziffer 1) vor, deren Wert nach dem unter a) entwickeltem Schema berechnet werden kann. Ein Algorithmus zur schnellen iterativen Berechnung von $y = x^n$ kann entsprechend dem folgendem Rechenschema konstruiert werden:

1	$\underline{x}$		$35 = n$	0	0	Initialisierung

y	*x*	*i mod 2* (= $z_{\nu-1}$)	*i div 2*	ν (Iterationsschritt)	μ (Multiplikation)	
$1 \cdot \underline{x}$	$\underline{x}^2$	1	17	1	1	$B(i) \equiv i > 0$
$\underline{x} \cdot \underline{x}^2$	$(\underline{x}^2)^2$	1	8	2	2	dto.
$\underline{x} \cdot \underline{x}^2$	$((\underline{x}^2)^2)^2$	0	4	3	2	dto.
$\underline{x} \cdot \underline{x}^2$	$(((\underline{x}^2)^2)^2)^2$	0	2	4	2	dto.
$\underline{x} \cdot \underline{x}^2$	$((((\underline{x}^2)^2)^2)^2)^2$	0	1	5	2	dto.
$\underline{x} \cdot \underline{x}^2 \cdot \underline{x}^{32}$		1	0	6	3	dto.
						$\neg B(i) \equiv i = 0$

Das Verfahren wird auch als *Square-and-Multiply* bezeichnet und z.B. durch den folgenden Block realisiert:

Block *x_hoch_n_iter (* **ein**: $x \in \mathbf{R}$, $n \in \mathbf{N}_0$; **aus**: $y \in \mathbf{R}$ *);*
    Daten $i \in \mathbf{N}_0$;
{ $y := 1.; i := n;$!! Z1: Vorbedingung P(y, x)
    Solange $i > 0$ **führe aus**
    { !! Z2: Laufbedingung B(i)
        Falls $i \bmod 2 \neq 0$!! Z3: Faktor $\underline{x}^{2^\nu}$ gefunden
        dann $y := y \cdot x;$!! Z4: 1. Schleifeninvariante I1(y, x)
        $i := i \operatorname{div} 2;$!! Z5: Berechnung der naechsten Dualziffer
        $x := x \cdot x;$!! Z6: 2. Schleifeninvariante I2(x)
    } !! Z7: Nachbedingung Q(y, x)
}

Für die Verifikation werden noch folgende Bezeichner eingeführt:

dld(x) Ganzzahliger Logarithmus von x zur Basis 2. Z.B. dld(35) = 6 (vgl. auch Aufg. 4.12). Damit kann die Anzahl d der Dualziffern (= Anzahl der Iterationen) von n berechnet werden.

ν Indiziert die Iterationsschritte. ν = 0 indiziert die Anfangswerte vor dem ersten Durchlauf. Man beachte, dass die Zusicherungen stets nach Ausführung der betreffenden Anweisung gelten.

μ Zählt fiktiv die Zahl der Faktoren (Multiplikationen) der Form $\underline{x}^{2^\nu}$. Es gilt offenbar $0 \le \mu \le g(n)$. Dabei bezeichnet g(n) die Zahl von Einsen in der Dualzahldarstellung von n. Es ist z.B. $g(35_{10}) = g(100011_2) = 3$.

$z_\nu \in \{0, 1\}$, $0 \le \nu < dld(n)$: ν-te Dualziffer von n (Zählung beginnt mit 0!). Im Beispiel $n = 35$ ist $z_0 = z_1 = z_5 = 1$.

Zusicherungen:

Z1: $P(y, x) \equiv (y_0 = \underline{x}^0 ; x_0 = \underline{x}^{2^0} = \underline{x})$

Z2: $B(i) \equiv (i > 0) = w \Rightarrow$ Exponent n noch nicht abgearbeitet. $\Rightarrow$ In n können noch 2er-Potenzen (Dualziffer 1) vorhanden sein.

Z3: $i_\nu \bmod 2 = z_{\nu-1} \Rightarrow (z_{\nu-1} = 1) = w \Rightarrow$ Dualziffer $z_{\nu-1} = 1$, $\nu > 0$.

Z4: $I1(y, x) \equiv (y_\nu = y_{\nu-1} \cdot x_{\nu-1}^{i_\nu \bmod 2} = y_{\nu-1} \cdot x_{\nu-1}^{z_{\nu-1}})$;

Z5: $i_\nu = i_{\nu-1} \operatorname{div} 2$

Z6: $I2(x) \equiv (x_\nu = x_{\nu-1} \cdot x_{\nu-1} = \underline{x}^{2^\nu})$ (Kann durch vollst. Induktion leicht bewiesen werden.)

Z7: $\neg B(i) \equiv ((i = 0) = w) \Rightarrow (\nu = dld(n) = d) = w \Rightarrow$ Alle Dualziffern im Exponent abgearbeitet.

$$(\mathrm{I1}(y,x) \wedge \neg \mathrm{B}(i)) \equiv (\, y_d = y_{d-1} \cdot x_{d-1}^{i_d \bmod 2} = y_{d-1} \cdot x_{d-1}^{z_{d-1}} \,)$$

$\Rightarrow$ Nachbedingung: $\quad \mathrm{Q}(y,x) \equiv (\, y_d = \prod_{\nu=0}^{d-1} (\underline{x}^{2^\nu})^{z_\nu} \,)$

Beweis durch vollständige Induktion. Es wird die Schreibweise *x* exp *e* statt x^e verwendet.

Induktionsanfang: $d = 1$:

Z4, Z1, Z3 $\quad \Rightarrow \quad y_1 = y_0 \cdot (x_0)^{z_0} = 1 \cdot \underline{x}^{z_0} = (\underline{x}^{2 \exp 0})^{z_0} = (\underline{x})^{z_0}$. (*)

Z4, Z6 $\quad \Rightarrow \quad y_2 = y_1 \cdot (x_1)^{z_1} = \underline{x}^{z_0} \cdot (\underline{x}^{2 \exp 1})^{z_1} = (\underline{x}^1)^{z_0} (\underline{x}^2)^{z_1}$.

Induktionsannahme:

$$y_{d-1} = \prod_{i=0}^{d-2} (\underline{x}^{2 \exp \nu})^{z_i} \qquad (**)$$

Dann gilt mit Z4 und (**):

$$y_d = y_{d-1}(x_{d-1})^{z_{d-1}} = \prod_{\nu=0}^{d-2} (\underline{x}^{2 \exp \nu})^{z_\nu} \cdot (\underline{x}^{2 \exp(d-1)})^{z_{d-1}} = \prod_{\nu=0}^{d-1} (\underline{x}^{2 \exp \nu})^{z_\nu} = (\underline{x}^1)^{z_0} (\underline{x}^2)^{z_1} (\underline{x}^4)^{z_2} \ldots (\underline{x}^{2 \exp(d-1)})^{z_{d-1}}$$

q.e.d..

c) Für die rekursive Lösung des Problems wird zunächst die Rechenvorschrift anders formuliert. Das Prinzip wird am Zahlenbeispiel $\mathrm{y}(x,35) = x^{35}$ erläutert:

$$\mathrm{y}(x,35) = \mathrm{y}(x^2,17) \cdot x = \mathrm{y}((x^2)^2,8) \cdot x \cdot x^2 = \mathrm{y}(((x^2)^2)^2,4) \cdot x \cdot x^2 = \mathrm{y}((((x^2)^2)^2)^2,2) \cdot x \cdot x^2$$
$$\mathrm{y}(x,35) = \mathrm{y}(((((x^2)^2)^2)^2)^2,1) \cdot x \cdot x^2 = \mathrm{y}(x^{32},1) \cdot x \cdot x^2 = x^{32} \cdot x^2 \cdot x.$$

Verallgemeinert gilt

$$\mathrm{y}(x,n) = \begin{cases} x & \text{falls } n = 1 \\ \mathrm{y}(x \cdot x, n \operatorname{div} 2) & \text{falls } n > 1, \text{n gerade} \\ \mathrm{y}(x \cdot x, n \operatorname{div} 2) \cdot x & \text{falls } n > 1, \text{n ungerade} \end{cases} .$$

Der folgende Block realisiert die Rechenvorschrift rekursiv und berücksichtigt auch den Fall n = 0.

```
Block x_hoch_n_rek ( ein: x ∈ R, n ∈ N0; aus: y ∈ R );
{       Falls  n = 0
        dann y := 1;
        sonst Falls  n = 1
                dann y := x;
                sonst Falls  i mod 2 = 0;
                        dann x_hoch_n_rek ( ein: x · x, n div 2; aus: y );
                        sonst {  x_hoch_n_rek ( ein: x · x, n div 2; aus: y);
                                 y := y · x;
                              }
```

```
}
```