Musterlösung

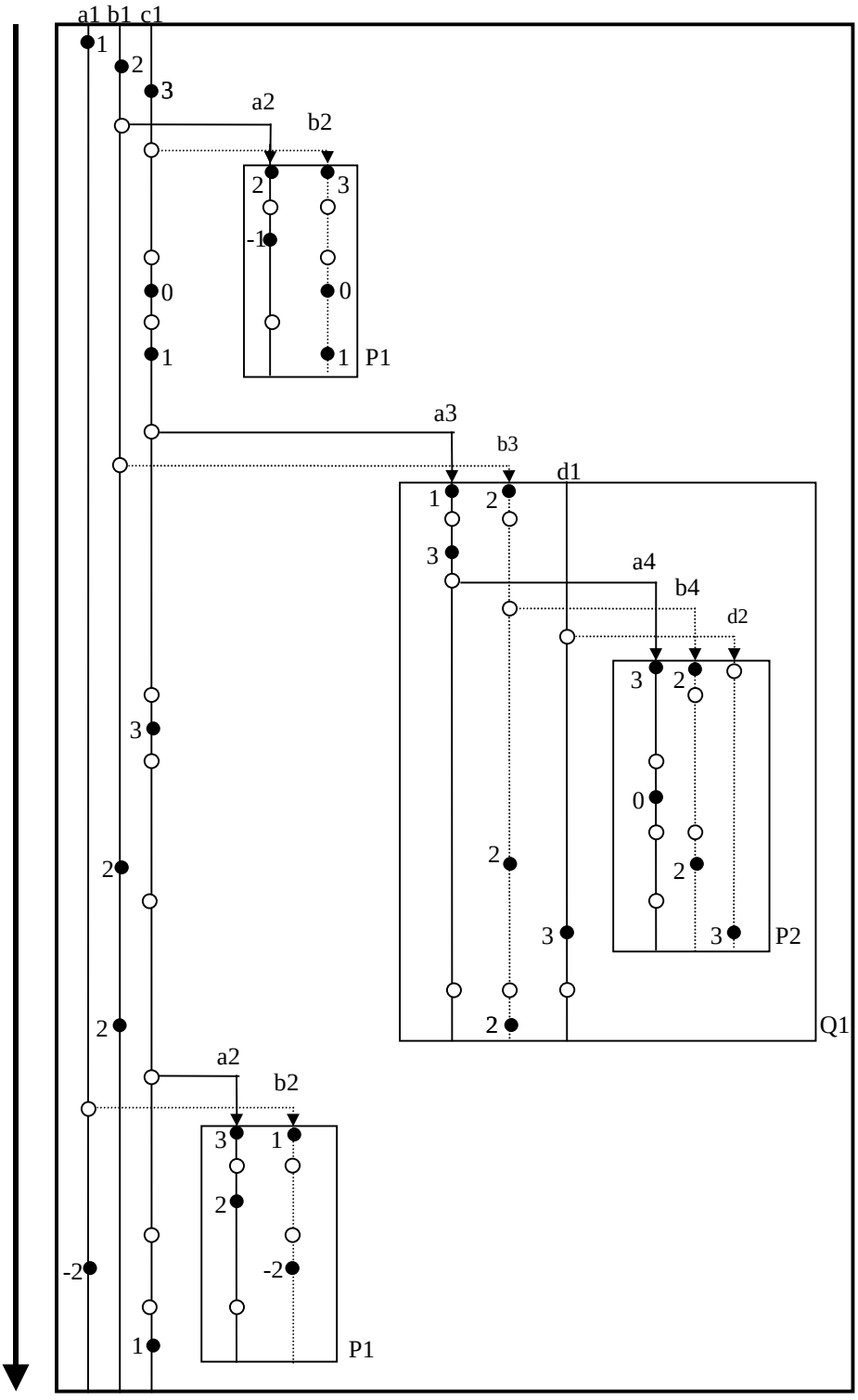
Übung 5

Aufgabe 5.1: Gültigkeitsbereich und Lebensdauer

(Korrigierter) Programmtext mit entsprechend ihrem Gültigkeitsbereich indizierten Variablen und Prozeduren:

```
MODULE M EXPORTS Main;

(* Beginn Gültigkeitsbereich 1 *)

VAR a1, b1, c1: INTEGER;

(* Beginn Gültigkeitsbereich 2 *)

PROCEDURE P1( a2: INTEGER; VAR b2: INTEGER) =
BEGIN
  a2 := a2 - b2;
  b2 := b2 - c1;
  c1 := c1 - a2;
END P1;

(* Ende Gültigkeitsbereich 2 *)

(* Beginn Gültigkeitsbereich 3 *)

PROCEDURE Q1 ( a3: INTEGER; VAR b3: INTEGER)=

(* Beginn Gültigkeitsbereich 4 *)

  PROCEDURE P2( a4: INTEGER; VAR b4: INTEGER; VAR d2: INTEGER) =
  BEGIN
    c1 := c1 + b4;
    a4 := a4 - c1;
    b4 := b4 + a4;
    d2 := c1 + a4;
  END P2;

(* Ende Gültigkeitsbereich 4*)

VAR d1 : INTEGER;
BEGIN
  a3 := a3 + b3;
  P2(a3, b3, d1);
  b3 := b3 + a3 - d1;
END Q1;

(* Ende Gültigkeitsbereich 3 *)

BEGIN
  a1 := 1;  b1 := 2;  c1 := 3;  P1( b1, c1);  Q1( c1, b1);  P1( c1, a1);
END M.
```

Lebensdauerdiagramm:



Aufgabe 5.2: Rekursion, Iteration

a) rekursiv

```
MODULE Potenz EXPORTS Main;

(* Ausgabe von b^n fuer Eingabe b und n
   Autor    : Antje Nowack
   Umgebung : PM3, Windows 95
   Erstellt : 15.11.2000   Letzte Aenderung: 15.11.2000
*)

IMPORT SIO;

(* Die folgenden (Hilfs-)Prozedur wird fuer die Ein- und
Ausgabe benutzt *)

PROCEDURE Eingabe() =

VAR b, n : REAL;

BEGIN
  SIO.PutText("Bitte geben Sie die Basis ein: ");
  b := SIO.GetReal();
  SIO.Nl();
  SIO.PutText("Bitte geben Sie den Exponenten ein: ");
  n := SIO.GetReal();
  SIO.Nl();
  SIO.PutText("Das Ergebnis lautet: ");
  SIO.PutReal(Potenz(b,n));
  SIO.Nl();
END Eingabe;

(* Die folgende Funktion gibt zu einer natuerlichen Zahl
an, ob sie gerade ist. *)

PROCEDURE IsEven (n : REAL) : BOOLEAN  =
BEGIN
  IF n = 0.0 THEN RETURN TRUE;
  ELSIF n = 1.0 THEN RETURN FALSE;
  ELSE RETURN (IsEven(n-2.0));
  END;
END IsEven ;

(* Die folgende Funktion liefert zu einer
Zahl ihr Quadrat. *)

PROCEDURE Quadrat(x : REAL) : REAL =
BEGIN
  RETURN x*x;
END Quadrat;

(* Die folgende Funktion gibt b^n fuer Eingabe b und n
aus. Sie ist rekursiv definiert. Die Idee dahinter ist
(b^(n/2))^2 = b^n fuer gerade n. Fuer n ungerade gilt:
b^n = b * (b^(n-1)) und n-1 ist gerade. Auf die Potenz
mit (n-1) im Exponenten kann die Regel fuer gerade n
benutzt werden. *)

PROCEDURE Potenz(b, n: REAL) : REAL =
BEGIN
  IF n=0.0 THEN
    RETURN 1.0;
  ELSIF IsEven(n) THEN

(* Rekursiver Aufruf im Fall, dass n gerade ist *)
```

```
    RETURN Quadrat(Potenz(b, n/2.0));
  ELSE

(* Rekursiver Aufruf im Fall, dass n ungerade ist *)

    RETURN (b * Potenz(b, n -1.0));
  END;
END Potenz;

BEGIN
  Eingabe();
END Potenz.
```

Bemerkung: Da $(b^2)^{n/2} = (b^{n/2})^2$ ist es egal, ob Quadrat(Potenz(b, n/2)) oder Potenz(Quadrat(b),n/2) benutzt im rekursiven Aufruf wird.

b) iterativ

```
MODULE Potenz EXPORTS Main;

(* Ausgabe von b^n fuer Eingabe b und n
   Autor    : Antje Nowack
   Umgebung : SRC-Modula-3 rel. 3.6, Windows 95
   Erstellt : 15.11.2000
   Letzte Aenderung: 15.11.2000
*)

IMPORT SIO;

(* Die folgende (Hilfs-)Prozedur wird fuer die Ein- und
Ausgabe benutzt *)

PROCEDURE Eingabe() =

VAR a, b, n : REAL;

BEGIN
  SIO.PutText("Bitte geben Sie die Basis ein: ");
  b := SIO.GetReal();
  SIO.Nl();
  SIO.PutText("Bitte geben Sie den Exponenten ein: ");
  n := SIO.GetReal();
  SIO.Nl();
  SIO.PutText("Das Ergebnis lautet: ");
  Potenz_iter(b, n, a);
  SIO.PutReal(a);
  SIO.Nl();
END Eingabe;

(*  Die  folgende  iterative  Prozedur  gibt  zu  einer  (natürlichen  Zahl,
einzugeben als Real) Zahl an, ob sie gerade ist. *)

PROCEDURE IsEven (n : REAL) : BOOLEAN  =
BEGIN
  WHILE n >= 0.0 DO
    IF n = 0.0 THEN RETURN TRUE;
    ELSIF n = 1.0 THEN RETURN FALSE;
    ELSE
      n := n – 2.0;
    END;
  END;
END IsEven ;

(* Die folgende Funktion liefert zu einer
Zahl ihr Quadrat. *)
```

```
PROCEDURE Quadrat(x : REAL) : REAL =
BEGIN
  RETURN x*x;
END Quadrat;

(* Die folgende Prozedur gibt b^n fuer Eingabe b und n
aus. Sie arbeitet iterativ. Die Idee dahinter ist
(b^(n/2))^2 = b^n fuer gerade n. Fuer n ungerade gilt:
b^n = b * (b^(n-1)) und n-1 ist gerade. Auf die Potenz
mit (n-1) im Exponenten kann die Regel fuer gerade n
benutzt werden.
Das Ergebnis wird in a akkumuliert.*)

PROCEDURE Potenz_iter(b, n: REAL; VAR a: REAL) =
VAR m, pot : REAL;
BEGIN

(* Hilfsvariablen werden initialisiert. *)
(* m haelt den aktuellen Exponenten *)

  m := n;

(* pot haelt die aktuelle Basis *)

  pot := b;

(* Die Ausgabe wird mit 1 initialisiert. Daher findet keine separate
Abfrage statt, ob der Exponent gleich 0 ist. *)

  a := 1.0;

(* While-Schleife wird im Fall, dass der Exponent nicht 0 ist ausgefuehrt
*)

  WHILE m > 0.0 DO

(* Wenn m gerade ist, wird die entsprechende Regel angewandt *)

    IF IsEven(m) THEN
      pot := Quadrat(pot);
      m := m/2.0;
    ELSE

(* ... ebenso im ungeraden Fall *)

      a := a * pot;
      m := m - 1.0;
    END;
  END;
END Potenz_iter;

BEGIN
 Eingabe();
END Potenz.
```

Aufgabe 5.3: Iteration

```
MODULE Fib EXPORTS Main;

(* Autor: Antje Nowack
   Umgebung: PM3, Windows95
   Erstellt: 22. 11. 2000
   Letzte Aenderung: 22. 11. 2000
*)
```

```
IMPORT SIO;

(* Die folgende Hilfsprozedur wird fuer die Ein-/Ausgabe benutzt. *)

PROCEDURE Eingabe() =

VAR a, b, n : REAL;

BEGIN
  SIO.PutText("Bitte geben Sie a ein: ");
  a := SIO.GetReal();
  SIO.Nl();
  SIO.PutText("Bitte geben Sie b ein: ");
  b := SIO.GetReal();
  SIO.Nl();
  SIO.PutText("Bitte geben Sie n ein: ");
  n := SIO.GetReal();
  SIO.Nl();
  SIO.PutText("Das Ergebnis lautet: ");
  SIO.PutReal(Fib(a, b, n);
  SIO.Nl();
END Eingabe;



PROCEDURE Fib (a, b, n : CARDINAL ) : CARDINAL =

(* a, b sind die zu uebergebenden Parameter, n ... *)

VAR first, second, result, number: CARDINAL;

(* first gibt das erste Argument auf Stufe number , second das zweite,
result haelt das Ergebnis auf der aktuellen Iterationsstufe *)

  BEGIN
    IF n = 0 THEN RETURN a;
    ELSIF n = 1 THEN RETURN b;
    ELSE

(* Initialisierung von first, second, number *)

      first := a;
      second := b;
      number := 2;

(* Der folgende Teil des Programms wird mehrmals durchlaufen *)
(* In der Schleife wird number pro Durchlauf um 1 erhoeht, first, second,
result entsprechend aktualisiert. Wenn number die Zahl n erreicht, wird der
Schleifendurchlauf beendet. *)

      WHILE number <= n DO
        result := b * first + a * second;
        first := second;
        second := result;
        number := number + 1;
      END;
      RETURN result;
    END;
  END Fib;

BEGIN
  Eingabe();
END Fib.
```

Für a = 1, b =1 liefert das Programm folgende Ergebnisse:

n	f(n)
0	1
1	1
2	2
3	3
4	5
5	8
6	13
7	21
8	34
9	55
10	89

Aufgabe 5.4: call-by-value, call-by-reference

a)

Beim Aufruf der Prozedur wird ein dem Typ des Wertparameters entsprechendes lokales Objekt angelegt. Ist der aktuelle Parameter eine Variable, so entsteht dabei eine Kopie des Parameter-Objekts. Veränderungen des formalen Parameters haben nur lokale Auswirkung. Der aktuelle Parameter bleibt unverändert. Bei Wertparametern bekommt die Prozedur nur den Wert zum Zeitpunkt des Aufrufs mitgeteilt.
Ein Referenzparameter hingegen wird beim Aufruf durch einen Verweis auf den aktuellen Parameter ersetzt. Jede Änderung des formalen Parameters ist direkt im aktuellen Parameter wirksam. Die Kommunikation ist in beiden Richtungen möglich.

Die (unüberlegte) Verwendung von Referenzparametern bringt meist Verständnisprobleme mit sich, da Änderungen potenziell Auswirkungen auf nicht offensichtlich zusammenhängende Argumente haben. Werden Referenzparameter in einer Funktion benutzt, so können hier außerhalb der Funktion Zustandsänderungen durchgeführt werden (vg. Seiteneffekt). Dies zerstört dies Lokalität einer Funktion. Die Änderung an Parametern in Prozeduren ist häufig eine Fehlerquelle, die schwer zu finden ist.

b)

```
MODULE Berechnung EXPORTS Main;
(* Autor: Antje Nowack
   Umgebung: PM3, Windows95
   Erstellt: 22. 11. 2000
   Letzte Aenderung: 22. 11. 2000
*)
IMPORT SIO, Text;


PROCEDURE Eingabe() =

VAR vokal, konsonant, gross, klein, doppel: CARDINAL;
    satz : TEXT;

BEGIN
  vokal := 0;
```

```
  konsonant := 0;
  gross := 0;
  klein := 0;
  doppel := 0;
  SIO.PutText("Geben Sie einen Satz ein: ");
  satz := SIO.GetLine();
  Berechnung (satz, vokal, konsonant, gross, klein, doppel);
  SIO.Nl();
END Eingabe;



(* Die folgende Prozedur berechnet zu einem eingegebenen Text die Anzahl
der eingegebenen Vokale, Konsonanten, Grossbuchstaben, Kleinbuchstaben,
Doppellaute. Die berechnenten befinden sich nach ausfuehrung der
Prozedur in vokal, konsonant, gross, klein, doppel *)

PROCEDURE Berechnung (string: TEXT; VAR vokal, konsonant, gross, klein,
doppel: CARDINAL ) =
  BEGIN

(* Gehe alle Zeichen des eingegebenen Textes durch. *)

    FOR i := 0 TO Text.Length(string)-1 DO

(* Fallunterscheidung nach Auswirkung der Zeichen auf das Ergebnis. *)

     CASE Text.GetChar(string, i) OF

(* Grossbuchstaben und Vokale und kein Doppellaut moeglich *)

       | 'I', 'O', 'U' =>
           vokal := vokal + 1;
           gross := gross +1;

(* Grossbuchstabe und Vokal und Doppellaute "Ai", "Au"  moeglich *)

       | 'A' =>
           vokal := vokal + 1;
           gross := gross +1;

(* Wenn auf das 'A' ein 'i' folgt und somit ein Doppellaut vorliegt ...*)

           IF i < Text.Length(string)-1 AND Text.GetChar(string, i+1)= 'i'
           THEN
             doppel := doppel + 1;

(* Wenn auf das 'A' ein 'u' folgt und somit ein Doppellaut vorliegt ...*)

           ELSIF i < Text.Length(string)-1 AND Text.GetChar(string, i+1)=
           'u' THEN
             doppel := doppel + 1;
           END;

(* Grossbuchstabe und Vokal und Doppellaut "Ei" moeglich *)

      | 'E' =>
           vokal := vokal + 1;
           gross := gross +1;

(* Wenn auf das 'E' ein 'i' folgt und somit ein Doppellaut vorliegt ...*)

           IF i < Text.Length(string)-1 AND Text.GetChar(string, i+1)= 'i'
           THEN
             doppel := doppel + 1;
           END;
```

```
(* Kleinbuchstaben und Vokale und kein Doppellaut moeglich *)

      | 'i', 'o', 'u' =>
          vokal := vokal + 1;
          klein := klein +1;

(* Kleinbuchstabe und Vokal und Doppellaute "ai", "au"  moeglich *)

      | 'a' =>
          vokal := vokal + 1;
          klein := klein +1;

(* Wenn auf das 'a' ein 'i' folgt und somit ein Doppellaut vorliegt ...*)

          IF i < Text.Length(string)-1 AND Text.GetChar(string, i+1)= 'i'
          THEN
            doppel := doppel + 1;

(* Wenn auf das 'a' ein 'u' folgt und somit ein Doppellaut vorliegt ...*)
          ELSIF i <= Text.Length(string)-2 AND Text.GetChar(string, i+1)=
          'u' THEN
            doppel := doppel + 1;
          END;

(* Grossbuchstabe und Vokal und Doppellaut "ei" moeglich *)

       | 'e' =>
          vokal := vokal + 1;
          klein := klein +1;

(* Wenn auf das 'e' ein 'i' folgt und somit ein Doppellaut vorliegt ...*)

          IF i < Text.Length(string)-1 AND Text.GetChar(string, i+1)= 'i'
          THEN
            doppel := doppel + 1;
          END;

(* Kleinbuchstaben und Konsonanten *)
       | 'b', 'c', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q',
         'r', 's', 't', 'v', 'w', 'x', 'y', 'z' =>
          klein := klein + 1;
          konsonant := konsonant +1;

(* Grossbuchstaben und Konsonanten *)

       | 'B', 'C', 'D', 'F', 'G', 'H', 'J', 'K', 'L', 'M', 'N', 'P', 'Q',
         'R', 'S', 'T', 'V', 'W', 'X', 'Y', 'Z' =>
          gross := gross + 1;
          konsonant := konsonant +1;

(* in allen anderen Faellen hat das Zeichen keinen Einfluss auf das
Ergebnis *)

       ELSE
     END;
    END;
    SIO.Nl();
    SIO.PutText("Anzahl der Kleinbuchstaben im Satz: ");
    SIO.PutInt(klein);
    SIO.Nl();
    SIO.PutText("Anzahl der Grossbuchstaben im Satz: ");
    SIO.PutInt(gross);
    SIO.Nl();
    SIO.PutText("Anzahl der Konsonanten im Satz: ");
    SIO.PutInt(konsonant);
    SIO.Nl();
    SIO.PutText("Anzahl der Vokale im Satz: ");
```

```
    SIO.PutInt(vokal);
    SIO.Nl();
    SIO.PutText("Anzahl der Doppellaute im Satz: ");
    SIO.PutInt(doppel);
  END Berechnung;

BEGIN
  Eingabe();
END Berechnung.
```

Testergebnisse:

Der erste Beispielsatz liefert folgendes Ergebnis:

Anzahl der Kleinbuchstaben im Satz: 43
Anzahl der Grossbuchstaben im Satz: 3
Anzahl der Konsonanten im Satz: 26
Anzahl der Vokale im Satz: 20
Anzahl der Doppellaute im Satz: 2

Der zweite Beispielsatz liefert folgendes Ergebnis:

Anzahl der Kleinbuchstaben im Satz: 37
Anzahl der Grossbuchstaben im Satz: 3
Anzahl der Konsonanten im Satz: 25
Anzahl der Vokale im Satz: 15
Anzahl der Doppellaute im Satz: 2