\autor\aufsa96\login964

Bausteine und Tools - im Unterricht noch immer vernachlässigt

Eberhard Lehmann

1. Einleitung

Unterrichtsbeobachtungen zeigen, daß das Programmieren mit Hilfsprogrammen und Bausteinen weiterhin sträflich vernachlässigt wird. Vielerorts erfolgt das Ausprogrammieren gleicher oder ähnlicher Teilalgorithmen immer wieder neu. Die Nachteile dieses Vorgehens liegen auf der Hand, z.B.:

- Verschwendung kostbarer Zeit durch unrationelles Arbeiten
- kein Vordringen zu komplexeren realitätsnahen Problemstellungen
- Vernachlässigen von in der Praxis heute überall gängigen Arbeitsweisen
- langweilige Unterrichtsphasen

Zunächst eine begriffliche Klärung:
Unter **(Programm-) Bausteinen** sollen hier Prozeduren aus Modulbibiotheken verstanden werden. Bausteine ersetzen das immer neue Entwerfen häufig wiederkehrender Algorithmen. Sie werden als Black-Box benutzt, jedoch müssen ihre Funktionen gut dokumentiert sein, um sie einfach benutzen zu können.

Unter **Tools** (Werkzeugen) sollen Hilfsprogramme verstanden werden, die - häufig ohne Verwendung einer Programmiersprache - beim Entwurfsprozeß helfen und damit viel Zeit einsparen, beispielsweise Maskengeneratoren. Für den Unterricht sind Tools besonders nützlich z.B. beim Oberflächendesign von Softwaresystemen.

Kehrt man die oben aufgezeigten Nachteile um, so ergeben sich Vorteile der Verwendung von Bausteinen und Tools.
- Zeitgewinn durch rationelles Arbeiten
- Möglichkeit des Vordringens zu komplexeren realitätsnahen Problemstellungen
- Vertrautmachen der Schüler mit einer heute überall gängigen Arbeitsweise
- Vermeidung langweiliger Programmierarbeit

2. Wo kommen die Bausteine her?

(1) Viele *Programmiersprachen enthalten Bausteine*. In Turbo-Pascal ist es z.B. die Unit Crt mit Prozeduren wie Clrscr und Gotoxy(a,b), also solchen mit und ohne Parameter.

(2) Von etlichen Fachkollegen wurden *Bausteinbibliotheken für die Unterrichtspraxis* angelegt. Hier wird z.B. auf Baumann [1] und Lehmann [2] verwiesen.

(3) Bei der *Analyse von Quelltexten* vorliegender Softwareprodukte lassen sich möglicherweise häufig benutzte Bausteine entdecken und in einer Bibliothek sammeln.

(4) Bei Problemlösungen sollte man immer darauf achten, ob sich Teilalgorithmen als Bausteine für andere Problemstellungen eignen. Auf diese Weise erweitert sich die Baustein-Formelsammlung.

3. Beispiele für die Verwendung von Tools und Bausteinen

Aufgabe: Es soll eine Oberfläche in Form einer Maske erzeugt werden, die der Eingabetafel von Gemüsesorten in der Gemüseabteilung eines Lebensmittelgeschäftes entspricht. Das Drücken einer Taste an der Eingabetafel zur Auswahl einer bestimmten Gemüsesorte soll durch Mausklick in der Maske simuliert werden.
Dieses Teilproblem aus einem größeren Projekt "Obst-Gemüse-Abteilung" kann so angegangen werden:

(1) Ein Tool - z.B. das Programm BILD_DEF aus [ ] - ermöglicht das Beschreiben des Bildschirms mit ASCII-Zeichen. Das Hilfsprogramm wird aufgerufen. Der Bildschirm kann beschrieben werden. Eine mitlaufende Spalten- und Zeilenangabe ermöglicht es uns, die Position der Optionstexte zu merken.

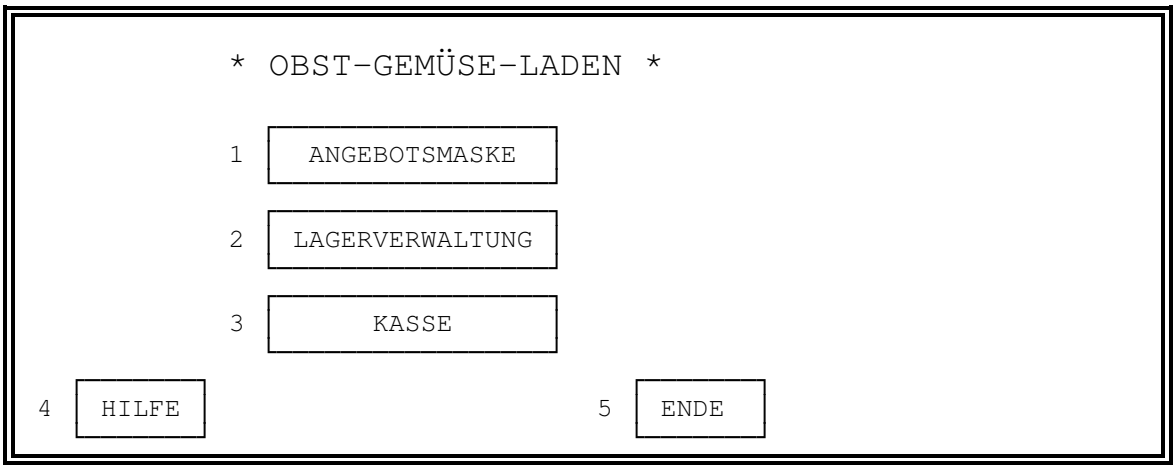


Bild 1

(2) Wir setzen nun für die Programmierung den Baustein "auswahl_pruefen(...)" ein. Sein Aufruf ermöglicht den Mausklick auf die Angebotsmaske (Option 1) oder die anderen Optionen und damit den Aufruf des zugehörigen Programmteils.

IF auswahlpruefen(33, 9,49, 9,ord('1')) THEN hp_option:=1; ***{Baustein}***

(3) Auch für die Erstellung der Angebotsmaske kann das Hilfsprogramm BILD_DEF eingesetzt werden. Ohne Gebrauch der Programmiersprache entsteht Bild 2.

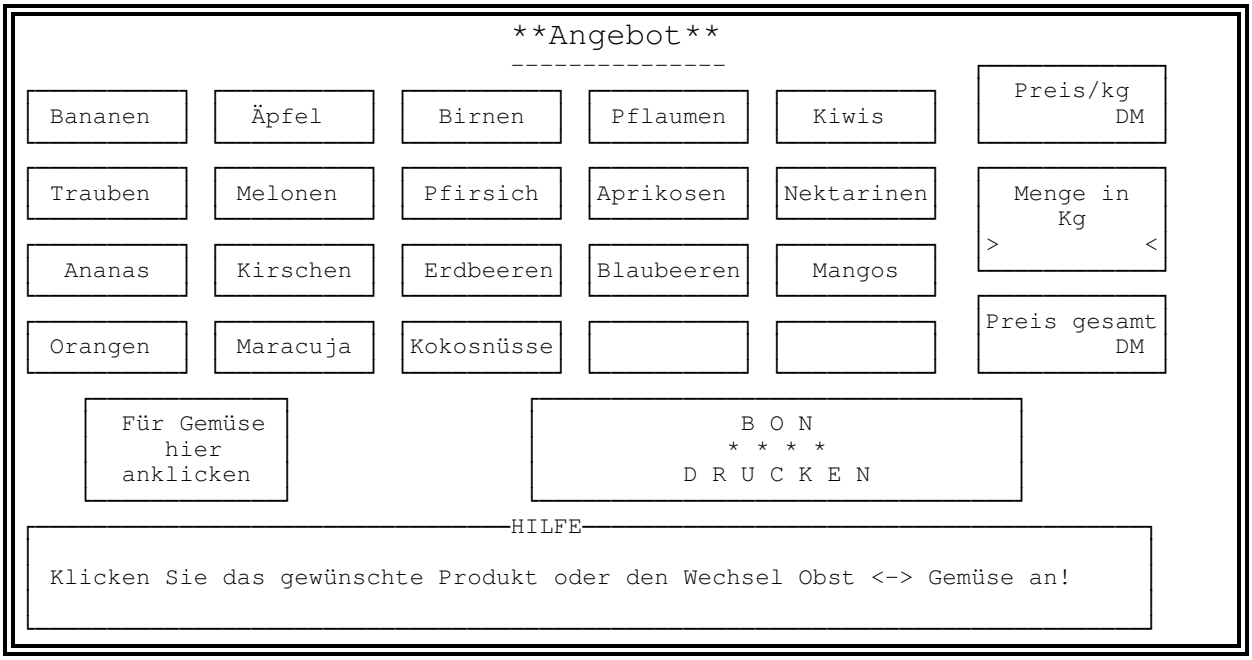


Bild 2

Der Aufruf der Maske innerhalb eines Turbo-Pascal-Programms wird durch die Prozedur *Textbild_aus(dateiname)* realisiert. Erneut lassen sich die einzelnen Obstsorten und die anderen Menüpunkte unter Verwendung von *auswahl_pruefen* durch Mausklick ansteuern.

Der folgende Programmauszug deutet an, in welchem Umfang bei dem Projekt mit Bausteinen gearbeitet wurde:

```
{-------------------------------------------------------------------------------------------------------------
-}
PROGRAM obst_HP;
Uses Crt,                                              {Standard-Unit}
   Ino_u,Balk_u,Bild_u,Maus_u,                         {Leh-Soft-Baustein-Units}
   obstda_u, obsth_u, obstm1_u,obstm2_u,obstm3_u;      {programmspezifische Units}
VAR    dateinam      : String[12];
        hp_option     : INTEGER;
       m2_option     : INTEGER;
       c             : CHAR;
       ok            : BOOLEAN;
{-------------------------------------------------------------------------------------------------------------
-}
PROCEDURE hilfe;
VAR c: CHAR;
```

```
BEGIN
  Gotoxy(1,24);                                          {Baustein}
  WRITELN('Hier werden später dem Geschäftsführer Hilfen und Infos angeboten.');
  WRITE('Zur Zeit wird hier der Bearbeitungsstand des Projekts gezeigt.');
  c:=Readkey;
  Output_textdatei(1,1,80,25,'obstinfo.pas','');                 {Baustein}
END;
{-------------------------------------------------------------------------------------------------------
-}
PROCEDURE in_der_hauptprogramm_maske_auswaehlen(VAR hp_option:INTEGER);
BEGIN
  IF auswahlpruefen(33, 9,49, 9,ord('1'))   THEN hp_option:=1;          {Baustein}
  IF auswahlpruefen(33,12,49,12,ord('2')) THEN hp_option:=2;          {Baustein}
  IF auswahlpruefen(33,15,49,15,ord('3')) THEN hp_option:=3;          {Baustein}
  IF auswahlpruefen(21,18,27,18,ord('4')) THEN hp_option:=4;          {Baustein}
  IF auswahlpruefen(56,18,62,18,ord('5')) THEN hp_option:=5;          {Baustein}

  CASE hp_option of
  1: BEGIN angebot END;                                {Modul/Unit obstm1_u.tpu}
  2: BEGIN pass(ok);
       if ok then lagerverwaltung(m2_option)           {Modul/Unit obstm2_u.tpu}
       else exit;
     END;
  3: BEGIN pass(ok);
       if ok then kasse                                {Modul/Unit obstm3_u.tpu}
       else exit;
     END;
  4: BEGIN pass(ok);
       if ok then hilfe
       else exit;
     END;
  5: BEGIN Clrscr; END;                                          {Baustein}
  END;
END;
{-------------------------------------------------------------------------------------------------------
-}
BEGIN  {Hauptprogramm}
  Clrscr;                                                        {Baustein}
  obstlogo;
  REPEAT
    dateinam:='obst_hp.mas';
    textbild_aus(dateinam);                                      {Baustein}
   {Laden der mit dem Hilfsprogramm BILD_DEF erstellten Bildschirmdatei}
    Mausodertaste;                                               {Baustein}
    hp_option:=0;
    in_der_hauptprogramm_maske_auswaehlen(hp_option);
    Clrscr;                                                      {Baustein}
```

```
  UNTIL (mtauswahl.taste=27) OR (hp_option=5);
  Clrscr;                                                    {Baustein}
END.
{-------------------------------------------------------------------------------------------------------------
-}
```

Mit *Output_textdatei(1,1,80,25,'obstinfo.pas','');* wurde ein weiterer wirkungsvoller Baustein eingesetzt. Er ermöglicht es ASCII-Dateien in den Programmlauf zu holen, beispielsweise für Hilfetexte oder andere Informationen. Diese können dann z.B. im Turbo-Pascal-Editor unabhängig vom Programm erstellt und ggf. geändert werden ohne das Programm neu kompilieren zu müssen [.

4. Bausteine dokumentieren - Baustein-Formelsammlung

Eine wichtige Voraussetzung für den Einsatz von Bausteinen an vielen Stellen ist ihre Dokumentation. Die Schüler sollten eine "Baustein-Formelsammlung" mit Kurzinformationen über die zur Verfügung stehenden Bausteine und ihre Funktion haben. **Im Unterricht muß ständig darauf geachtet werden, daß bei allen geeigneten Gelegenheiten Bausteine verwendet werden. Nur so wird schließlich die Verwendung von Bausteinen selbstverständlich.** Gelegentlich wird man einen Baustein auch öffnen und hineinsehen. Auch wird sich die Bausteinsammlung möglicherweie ständig erweitern.

Dokumentationsbeispiel:

```
PROCEDURE input_string(  x,y                 : INTEGER;
                         input_aufforderung  : STRING;
                         VAR input           : STRING;
                         stringlaenge        : INTEGER;
                         erlaubt             : Zeichenmenge);
```

Diese Prozedur liest einen String ein. Die Parameter bedeuten:
x: Spaltenposition, an dem der input-Aufforderungstext beginnt,
y: Zeilenposition
input_aufforderung: Text für die Inputaufforderung,
input: Der einzugebende String. Er muá vor Aufruf der Prozedur initialisiert werden!
stringlaenge: Die erlaubte Stringlänge,
erlaubt: Die erlaubte Zeichenmenge für den einzugebenden String.

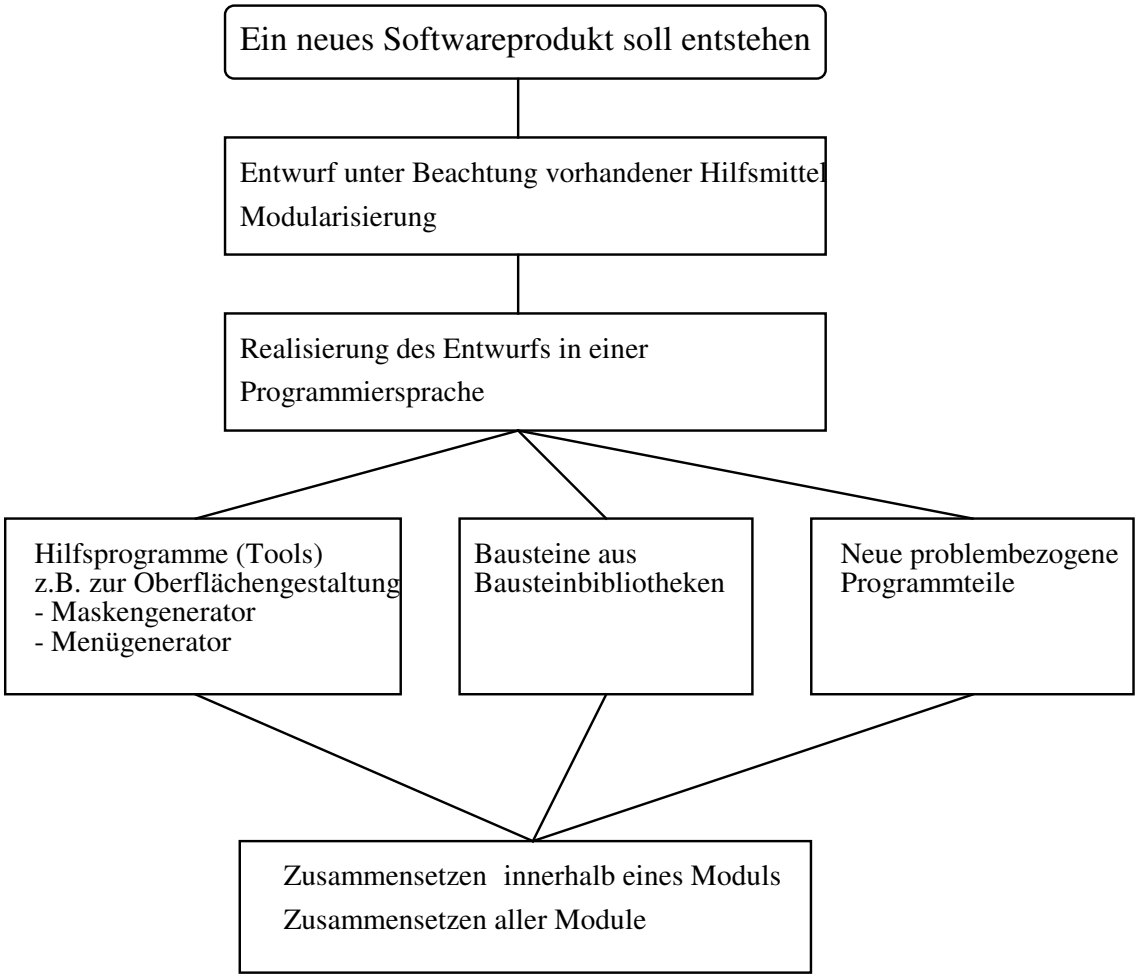
Besonders nützlich sind bei der Dokumentation von Bausteinen Programmierbeispiele:

```
PROGRAM nameneingeben;
USES  Crt; ino_u;
VAR i: INTEGER;
```

```
   name: ARRAY[1..10] OF STRING;
BEGIN
  Clrscr;
  FOR i:=1 TO 5 DO name[i]:='';
  FOR i:=1 TO 5DO
  Input_string(5+8*i,15+i,'Namen eingeben, Name = ',name[i],8,['A'..'Z']);
  Clrscr;
END.
```

5. Bearbeiten komplexer Problemstellungen mit Bausteinen

Die aus dem Informatikunterricht sattsam bekannten kleinen Programme zur Einführung neuer Begriffe einer Programmiersprache oder zur Übung haben kaum längerfristigen Wert - es sind "Wegwerfprogramme".



(Programmieren durch Konfigurieren)

Bild 3

Ungleich wirkungsvoller sind kleine Programme, die Tools oder Bausteine verwenden. Viele der verwendeten Anweisungen lösen nun für sich schon umfangreichere Aufgabenstellungen. Komplexere und realitätsnahe Problemstellungen können in vernünftiger Zeit (der Grundkurs Informatik hat schließlich in der Regel nur 3 Wochenstunden!) nur bei Verwendung von Tools und Bausteinen bewältigt werden. Bild 3 veranschaulicht die Stellung von Tools und Bausteinen innerhalb der Softwareerstellung.

6. Ein neuer Baustein entsteht

Wir zeigen diese Aspekte an einem Programm, das die Eingabe von Stundenplänen ermöglicht, die dann in einer Datei gespeichert werden und aus dieser wieder abgerufen werden können. Aus der Problemlösung erwächst ein neuer wiederverwendbarer Baustein "Stundenpläne" - einsetzbar innerhalb anderer umfangreicherer Aufgabenstellungen.
Die Bearbeitung erfolgt gemäß der groben (verkürzten) Darstellung von Bild 3. Wir verwenden hier einige Bausteine aus den Bibliotheken (Units) Crt, Ino_u, Balk_u. Nachdem sich das Stundenplanprogramm als lauffähig erwiesen hat, wird es als neuer Baustein (Unit Stunplan) zur Wiederverwendung abgelegt. Wir zeigen hier gleich die entstandene Unit und formulieren ein Testprogramm.

Hinweis: Die verwendeten Bausteine und der neu entstandene Baustein werden fett markiert. Die Prozeduren zum Speichern und Laden der Stundenplandatei sind nicht bzgl. des Dateihandling hier nicht abgesichert.

```
UNIT Stunplan                                          {der neue Baustein}
INTERFACE
USES Crt,Ino_u, Balk_u;
TYPE plan          = ARRAY[0..5,0..9] OF STRING[10];
     plan_datei    = FILE OF plan;
VAR  plandatei     : plan_datei;

PROCEDURE stundenplaene(VAR plandatei: plan_datei);
```

```
IMPLEMENTATION

 TYPE string12      = STRING[12];
 VAR  planstelle    : plan;
      dateiname     : string12;
      fall          : CHAR;
(*-------------------------------------------------------------------------------------------*)
```

```
PROCEDURE plan_indatei( VAR plandatei : plan_datei; planstelle: plan;
                        VAR dateiname : string12);
BEGIN
  Clrscr;
  dateiname:='Plan-1.pla';   { Voreinstellung}
  input_string(1,1,'Speicherung des Plans - Dateiname = ',
          dateiname,12,['A'..'Z','a'..'z','0'..'9','.','-']);
  ASSIGN(plandatei, dateiname);
  REWRITE(plandatei);
  WRITE(plandatei,planstelle);
  CLOSE(plandatei);
END;
(*-------------------------------------------------------------------------------------------*)
PROCEDURE plan_ausdatei( VAR plandatei: plan_datei;
                                VAR dateiname: string12);
VAR planstelle        : plan;
      weiter          : CHAR;
      i,j             : INTEGER;

BEGIN
  Clrscr;
  dateiname:='Plan-1.pla';   {Voreinstellung}
  input_string(1,1,'Ausgabe des Plans - Dateiname = ',
          dateiname,12,['A'..'Z','a'..'z','0'..'9','.','-']);
  ASSIGN(plandatei, dateiname);
  RESET(plandatei);
  READ(plandatei,planstelle);
  CLOSE(plandatei);
  Clrscr;
  WRITE('   Montag    Dienstag   Mittwoch');
  WRITE('   Donnerstag Freitag   Samstag');
  WRITELN; WRITELN;
  FOR i:=1 TO 10 DO WRITELN(i);
  FOR i:=0 TO 5 DO
  FOR j:=0 TO 9 DO
  BEGIN
    GOTOXY(4+i*11,3+j); WRITE(' ',planstelle[i,j]);
  END;
  weiter:=READKEY;
END;
(*-------------------------------------------------------------------------------------------*)
PROCEDURE plan_erstellen(VAR planstelle: plan);
VAR i,j                 : INTEGER;
```

```
      c,antwort          : CHAR;

BEGIN
  FOR i:=0 TO 5 DO
  FOR j:=0 TO 9 DO planstelle[i,j]:='';
  Clrscr;
  WRITE('   Montag    Dienstag   Mittwoch');
  WRITE('   Donnerstag Freitag   Samstag');
  WRITELN; WRITELN;
  FOR i:=1 TO 10 DO WRITELN(i);
  REPEAT
    FOR i:=0 TO 5 DO
    FOR j:=0 TO 9 DO
    input_string(4+i*11,3+j,' ',planstelle[i,j],10,
                 ['A'..'Z','a'..'z','0'..'9','.','-']);
    antwort:='?';
    input_zeichen(1,23,'Alles Ok ? (j,n) = ',antwort,['J','N','j','n']);
  UNTIL antwort IN ['j','J'];
  plan_indatei (plandatei, planstelle,dateiname);
END;
(*-------------------------------------------------------------*)
PROCEDURE stundenplaene(VAR plandatei: plan_datei);
BEGIN
  REPEAT
    Clrscr;
    Rahmen_zeichnen2(2,2,40,12,'','',green,green);
    Gotoxy(4,4); WRITE('1: Plan erstellen / speichern');
    Gotoxy(4,6); WRITE('2: Plan ausgeben');
    Gotoxy(4,8); WRITE('0: Ende');
    fall:='0';
    input_zeichen(4,10,'Wählen Sie 1,2 oder 0 / Eingabe = ', fall,['0','1','2']);
    CASE fall OF
      '1': plan_erstellen(planstelle);
      '2': plan_ausdatei(plandatei, dateiname);
      '0': Clrscr;
    END;
  UNTIL fall='0';
END;

BEGIN END.
```

Testprogramm:

```
PROGRAM stundenplaene_bearbeiten;  {Testprogramm für die Unit "Stunplan"}
USES Stunplan;
BEGIN
  stundenplaene(plandatei);
END.
```

Literatur:

[1] Baumann,R.: Elementare Computergrafik, Stuttgart: Klett, 1994 *(mit Grafikbausteinen)*

[2] Lehmann, E.: Programmieren in Turbo-Pascal mit Bausteinen, Bonn: Dümmler, 1994 *(mit Tools zur einfachen Masken- und Menügenerierung)*

[3] ders. : Projekte im Informatik-Unterricht, Analyse, Konstruktion, Wartung komplexer Software; Bonn: Dümmler, 1995

[4] Künzell, Lehmann, Matzanke: Grafische Darstellung der Baumrekursion, LOGIN 4´96 *(Arbeit mit Bausteinen aus [1] und [2])*