

struktex.sty*

Jobst Hoffmann
Fachhochschule Aachen, Abt. Jülich
Ginsterweg 1
52428 Jülich
Bundesrepublik Deutschland
gedruckt am 18. August 2006

Zusammenfassung

Dieser Artikel beschreibt den Einsatz und die Implementation der \LaTeX -*package* `struktex.sty` zum Setzen von Struktogrammen nach Nassi-Shneidermann.

Inhaltsverzeichnis

1	Vorwort	2	5	Verschiedene Beispieldateien	21
		5.1		Beispieldatei Nr. 1	21
		5.2		Beispieldatei Nr. 2	22
		5.3		Beispieldatei Nr. 3	23
		5.4		Beispieldatei Nr. 4	27
2	Hinweise zur Pflege und Installation sowie die Treiberdatei zur Erzeugung dieser Dokumentation	3			
3	Die Benutzungsschnittstelle	5	6	Makros zur Erstellung der Dokumentation des <code>struktex.sty</code>	29
	3.1 Spezielle Zeichen und Textdarstellung	6			
	3.2 Makros zur Darstellung von Variablen, Schlüsselwörtern und anderen programmierspezifischen Details	6	7	Makefile	34
	3.3 Die Makros zur Erzeugung von Struktogrammen	8	8	Stil Datei zur einfachen Eingabe von Struktogrammen beim Arbeiten mit dem (X)emacs und AUCT\TeX	38
4	Beipieldatei zum Einbinden in die Dokumentation	21	9	Die Implementation	42
			9.1	Allgemeines	42
			9.2	Diverse Makros zur Vorbereitung	44

*Diese Datei hat die Versionsnummer v8.2a, wurde zuletzt bearbeitet am 2006/01/20, und die Dokumentation datiert vom 2006/01/20.

9.3 Darstellung von Variablen etc.	45	9.5 Die einzelnen Makros . . .	50
9.4 Belegung von Registern	49	9.6 Zusätzliche Makros (Verifikation)	66

1 Vorwort

Mit dem hier beschriebenen Makropaket ist es möglich, Struktogramme mit \LaTeX zu zeichnen. Das Makropaket wird im folgenden immer \StukTeX genannt. Es ist in der Lage, die wichtigsten Elemente eines Struktogrammes wie z. B. Verarbeitungsblöcke, Schleifenkonstrukte, Sinnbilder für Alternativen usw. zu generieren. Die Struktogramme werden mit Hilfe der Picture-Umgebung von \LaTeX erzeugt.

Ab Version 4.1a werden die mathematischen Symbole von \AMS-TeX geladen, die den mathematischen Zeichensatz erweitern und andere Darstellungen von Mengensymbolen (etwa \mathbb{N} , \mathbb{Z} und \mathbb{R} für die natürlichen, ganzen und reellen Zahlen) ermöglichen. Insbesondere das Zeichen für die leere Menge (\emptyset) ist in der Darstellung auffälliger als das standardmäßige Zeichen („ \emptyset “) und somit besser für die Darstellung von Struktogrammen geeignet.

Weiterhin ist aus dem `oz.sty` die Idee übernommen, Variablenamen in *italics* zu setzen, ohne dass die teilweise unschönen Zwischenräume erzeugt werden.

Die Entwicklung dieses Makropaketes ist noch nicht abgeschlossen. Es war geplant, die Struktogramme unter dem Einsatz des Makros aus `emlines2.sty` zu zeichnen, um die durch \LaTeX gegebenen Einschränkungen – es gibt nur vordefinierte Steigungen – aufzuheben. Dies ist – für das `\ifthenelse` mit den Versionen 4.1a und 4.1b, für das `\switch` mit der Version 4.2a – erledigt, nicht jedoch für Systeme, die die entsprechenden `\special{...}`-Befehle nicht unterstützen. Erreicht werden kann dies jedoch durch Einsatz entsprechender Makros aus dem `curves.sty`. Seit der Version 8.0a wird das Paket `pict2e.sty` unterstützt, das mittels der üblichen Treiber die von der `picture`-Umgebung bekannten Beschränkungen auf nur wenige Steigungen im wesentlichen aufhebt, so dass sich die Benutzung der entsprechenden Option (s.u.) dauerhaft empfiehlt.

Ebenso ist es geplant, Struktogramme um Kommentarblöcke zu erweitern, wie sie in dem Buch von Futschek ([Fut89]) eingesetzt werden. Dieses ist ebenfalls mit der Version 8.0a realisiert worden.

Weitere Zukunftspläne sind:

1. Ein `\otherwise`-Zweig beim `\switch` (abgeschlossen durch die Version 4.2a).
2. Die Neuimplementation der `declaration`-Umgebung mittels der `list`-Umgebung gemäß [GMS94, Abs. 3.3.4] (abgeschlossen mit der Version 4.5a).
3. Die Anpassung an $\text{\LaTeX} 2_{\epsilon}$ im Sinne eines Packages (abgeschlossen durch die Version 4.0a).
4. Die Verbesserung der Dokumentation, um Teile des Algorithmus verständlicher zu machen.
5. Die Unabhängigkeit des `struktex.sty` von anderen `.sty`-Dateien wie etwa dem `JHfMakro.sty` (abgeschlossen mit der Version 4.5a).
6. die vollständige Implementation der Makros `\pVar`, `\pKey`, `\pFonts`, `\pTrue`, `\pFalse` und `\pBoolValue` (erledigt vor Version 7.0),

7. die vollständige Internalisierung von Kommandos, die nur in der Umgebung **struktogramm** Sinn machen. Internalisierung bedeutet, dass diese Kommandos nur innerhalb der Umgebung definiert sind. Dies hat den Zweck, das Paket mit anderen Paketen verträglicher zu gestalten, etwa mit dem `ifthenelse.sty`. Begonnen wurde die Internalisierung mit der Version 4.4a.
8. Die Unabhängigkeit der Dokumentation von anderen `.sty`-Dateien wie etwa dem `JHfMakro.sty` (abgeschlossen mit der Version 5.0).
9. Eine alternative Darstellung der Deklarationen, wie sie von Rico Bolz vorgeschlagen wurde
10. Wiedereinführung der `make`-Ziele `dist-src dist-tar` und `dist-zip`.

Der derzeitige Stand der Implementierung ist an entsprechender Stelle vermerkt.

2 Hinweise zur Pflege und Installation sowie die Treiberdatei zur Erzeugung dieser Dokumentation

Das Paket, zu dem der `struktex.sty` gehört, besteht aus insgesamt sechs Dateien:

```

LIESMICH,
README,
struktex.ins,
struktex.dtx,
struktex.de.pdf und
struktex.en.pdf.

```

Um daraus einerseits die Dokumentation, andererseits die `.sty`-Datei zu erzeugen, muss folgendermaßen vorgegangen werden:

Zunächst wird mit z. B.

```
latex struktex.ins
```

die Datei `struktex.ins` formatiert. Dieser Formatierungslauf erzeugt elf weitere Dateien. Dies sind zunächst die drei `.sty`-Dateien `struktex.sty`, `struktexf.sty` und `struktexp.sty`, die beim Einsatz des `struktex.sty` benötigt werden; weiterhin sind es die beiden Dateien `struktex.test_0.nss` und `struktex.doc.sty`, die zur Erzeugung der hier vorliegenden Dokumentation benötigt werden. Dazu kommen drei Testdateien `struktex.test_i.nss`, $i = 1(2)3$, sowie die beiden Dateien `struktex.makemake` und `struktex.mk` (vgl. Abschnitt 7).

Die Dokumentation wird wie üblich durch

```

latex struktex.dtx
latex struktex.dtx
makeindex -s gind.ist struktex.idx
latex struktex.dtx

```

erzeugt.¹ Das Ergebnis dieses Formatierlaufes ist die Dokumentation in Form einer .dvi-Datei, die in gewohnter Weise weiterbearbeitet werden kann. Weitere Informationen zum Arbeiten mit der integrierten Dokumentation findet man in [Mit01] und [MDB01]. Die Dateien `tst_strf.tex`, `tst_strp.tex` schließlich sind Dateien zum Austesten der hier beschriebenen Makros.

Die Installation wird abgeschlossen, indem die Datei `struktex.sty` in ein Verzeichnis verschoben wird, das von \TeX gefunden werden kann, das ist in einer TDS-konformen Installation typischerweise `.../tex/latex/struktex/`, die Dokumentation wird analog in das Verzeichnis `.../doc/latex/struktex/` verschoben.²

Sollen Änderungen durchgeführt werden, so sollten neben diesen die Werte von `\fileversion`, `\filedate` und `\docdate` bei Bedarf entsprechend geändert werden. Weiterhin sollte darauf geachtet werden, dass die Revisionsgeschichte durch Einträge von

$$\backslash\text{changes}\{\langle\textit{Version}\rangle\}\{\langle\textit{Datum}\rangle\}\{\langle\textit{Kommentar}\rangle\}$$

weitergeschrieben wird. $\langle\textit{Version}\rangle$ gibt die Versionsnummer an, unter der die jeweilige Änderung durchgeführt wurde, $\langle\textit{Datum}\rangle$ gibt das Datum in der Form `yy/mm/dd` an und $\langle\textit{Kommentar}\rangle$ erläutert die jeweilige Änderung. $\langle\textit{Kommentar}\rangle$ darf nicht mehr als 64 Zeichen enthalten. Daher sollten Kommandos nicht mit dem „\“ (*backslash*) eingeleitet werden, sondern mit dem „“ (*accent*).

Die nächsten Anweisungen bilden den Treiber für die hier vorliegende Dokumentation.

```

1 \documentclass[a4paper, english, ngerman]{ltxdoc} %swap english and ngerman
2                                     % for producing an english version of this text
3
4 \usepackage{babel}                  % for switching the documentation language
5 \usepackage{struktoc}               % the style-file for formatting this
6                                     % documentation
7 \usepackage[pict2e, % <----- to produce finer results
8                                     % visible under xdvi, alternatives are
9                                     % curves or emlines2 (visible only under
10                                    % ghostscript), leave out if not
11                                    % available
12   verification]
13   {struktex}
14 \usepackage{url}
15 \GetFileInfo{struktex.sty}
16
17 \EnableCrossrefs
18 %\DisableCrossrefs % say \DisableCrossrefs if index is ready
19
20 %\RecordChanges % say \RecordChanges to gather update information
21
22 %\CodelineIndex % say \CodelineIndex to index entry code by line number
23
24 \OnlyDescription % say \OnlyDescription to omit the implementation details

```

¹Die Erzeugung der Dokumentation kann durch den Einsatz einer `make`-Datei vereinfacht werden, vgl. Abschnitt 7

²Wenn die automatische Installation (vgl. Abschnitt 7) vorgenommen wird, erfolgt diese in die Verzeichnisse `.../doc/latex/jhf/struktex/`, `.../tex/latex/jhf/struktex/` und `.../source/latex/jhf/struktex/`.

```

25
26 \MakeShortVerb{\|} % |\foo| acts like \verb+\foo+
27
28 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
29 % to avoid underfull ... messages while formatting two/three columns
30 \hbadness=10000 \vbadness=10000
31
32 \def\languageNGerman{3}
33
34 \begin{document}
35 \makeatletter
36 \@ifundefined{selectlanguageEnglish}{}{\selectlanguage{english}}
37 \makeatother
38 \DocInput{struktex.dtx}
39 \end{document}

```

3 Die Benutzungsschnittstelle

Der `struktex.sty` wird wie jede andere `.sty`-Datei als *package* in ein \LaTeX -Dokument eingebunden:

```
\usepackage[Optionen]{struktex}
```

Die folgenden Optionen stehen zur Verfügung:

1. `emlines`, `curves` oder `pict2e`:

Durch Angabe einer der drei Optionen ist es möglich, beliebige Steigungen in Struktogrammen zu zeichnen. Erstere Option ist sinnvoll, wenn mit dem $\text{emT}_{\text{E}}\text{X}$ -Paket von Eberhard Mattes gearbeitet wird (DOS oder OS/2), ansonsten wird der Einsatz von `pict2e` empfohlen. Der Einsatz des Paketes `curves.sty` (Ian Maclaine-cross), das das Zeichnen von Geraden beliebiger Steigungen durch das Setzen vieler einzelner Punkte ermöglicht, ist durch das Erscheinen des Paketes `pict2e.sty` (Hubert Gäßlein und Rolf Niepraschk) im Prinzip obsolet, aus Kompatibilitätsgründen wird er weiter unterstützt. Durch die Angabe einer der genannten Option wird das jeweilige Paket (`emlines2.sty`, `curves.sty` bzw. `pict2e.sty`) automatisch geladen.

2. `verification`:

Nur wenn diese Option gesetzt ist, steht `\assert` als Kommando zur Verfügung.

3. `nofiller`:

Setzen dieser Option lässt jeden Freiraum leer. Ansonsten wird Freiraum in Alternativen als \emptyset markiert.

4. `draft`, `final`:

Diese Optionen dienen in üblicher Weise dazu, den Entwurf beziehungsweise die endgültige Fassung zu kennzeichnen (vgl. `\sProofOn`/`\sProofOff`). Im Entwurfsmodus werden die vier Eckpunkte eines Struktogramms in der vom Benutzer vorgegebenen Größe ausgegeben, diese Markierung fällt in der endgültigen Fassung weg.

Nach dem Laden der `.sty`-Datei stehen verschiedene Kommandos und Umgebungen zur Verfügung, die das Zeichnen der Struktogramme ermöglichen.

`\StrukTeX` Zunächst sei der Befehl erwähnt, der das Logo `StrukTeX` erzeugt:

```
\StrukTeX
```

Damit kann in Dokumentationen auf die hier vorliegende Stil-Option verwiesen werden.

3.1 Spezielle Zeichen und Textdarstellung

`\nat` Wegen ihres häufigen Auftretens sind die Mengen der natürlichen, ganzen, reellen
`\integer` und komplexen Zahlen (\mathbb{N} , \mathbb{Z} , \mathbb{R} und \mathbb{C}) im Mathematik-Modus über die folgenden
`\real` Makros erreichbar: `\nat`, `\integer`, `\real` und `\complex`. Ebenso ist das mit
`\complex` `\emptyset` erzeugte „ \emptyset “ als Zeichen für die leere Anweisung auffälliger als das
`\emptyset` standardmäßige Zeichen „ \emptyset “. Andere Mengensymbole wie \mathbb{L} (für Lösungsmenge)
sind über `\mathbb{L}` zu erzeugen.

`\MathItalics` Mit diesen beiden Makros kann die Darstellung von Variablenamen beeinflusst
`\MathNormal` werden:

```

NeuerWert = AlterWert + Korrektur
\MathNormal
\[
NeuerWert = AlterWert + Korrektur
\]
```

und

```

NeuerWert = AlterWert + Korrektur
\MathItalics
\[
NeuerWert = AlterWert + Korrektur
\]
```

3.2 Makros zur Darstellung von Variablen, Schlüsselwörtern und anderen programmierspezifischen Details

`\pVariable` Mit `\pVariable{<Variablenname>}` wird ein Variablenname gesetzt. `<Variablenname>`
`\pVar` ist dabei ein Bezeichner eine Variable, wobei der Unterstrich „_“, das kaufmänni-
`\pKeyword` sche Und „&“ und das Dach „^“ als Teile des Variablenamens erlaubt sind:
`\pKey`

```

\pComment cEineNormaleVariable
c_eine_normale_Variable
&iAdresseEinerVariablen
zZeigerAufEineVariable^.sInhalt
\obeylines
\renewcommand{\pLanguage}{C}
\pVariable{cEineNormaleVariable}
\pVariable{c_eine_normale_Variable}
\pVariable{&iAdresseEinerVariablen}
\renewcommand{\pLanguage}{Pascal}
\pVariable{zZeigerAufEineVariable^.sInhalt}
```

Leerzeichen werden beachtet, so dass ganze Anweisungen geschrieben werden können. Es darf als Abkürzung `\pVar` benutzt werden.

Entsprechend wird mit `\pKeyword{<Schlüsselwort>}` ein Schlüsselwort gesetzt. Dabei ist *<Schlüsselwort>* ein Schlüsselwort in einer Programmiersprache, wobei der Unterstrich „_“ und das *hash*-Zeichen „#“ als Teil des Schlüsselwortes erlaubt ist. Damit lässt sich setzen:

```
begin                \obeylines
program             \pKeyword{begin}
#include            \renewcommand{\pLanguage}{Pascal}
                   \pKeyword{program}
                   \renewcommand{\pLanguage}{C}
                   \pKeyword{#include}
```

Auch `\pKeyword` darf abgekürzt werden: `\pKey`. Damit erzeugt dann der Quelltext

```
\renewcommand{\pLanguage}{Pascal}
\pKey{begin} \pExp{iVar := iVar + 1;} \pKey{end}
```

als Ausgabe dieses Ergebnis:

```
begin iVar := iVar + 1; end
```

In ähnlicher Weise dient `\pComment` zur Darstellung von Kommentar. Das Argument darf nur Zeichen der $\text{T}_{\text{E}}\text{X}$ -Kategorie *letter* haben, Zeichen, die einen Kommentar einleiten, müssen geschrieben werden. `\pComment` kann nicht abgekürzt werden. Beispielsweise ergibt

```
\pExp{a = sqrt(a);} \pComment{// Iteration}
```

die Zeile

```
a = sqrt(a); // Iteration
```

`\pTrue` Logische Werte spielen in der Programmierung eine wesentliche Rolle. Mit
`\pFalse` `\pTrue` und `\pFalse` sind entsprechende Werte vorgegeben: WAHR und FALSCH.
`\pFonts` Der Makro `\pFonts` dient der Auswahl von Fonts zur Darstellung von Varia-
`\pBoolValue` blen, Schlüsselwörtern und Kommentar:

```
\pFonts{<Variablenfont>}{<Schlüsselwortfont>} {<Kommentarfont>}
```

Vorbesetzt sind die einzelnen Fonts mit

- *<Variablenfont>* als `\small\sffamily`,
- *<Schlüsselwortfont>* als `\small\sffamily\bfseries` und
- *<Kommentarfont>* als `\small\sffamily\slshape`.

Damit wird die obige Zeile nach

```
\pFonts{\itshape}{\sffamily\bfseries}{\scshape}
\pVar{a = }\pKey{sqrt}\pVar{a);} \pComment{// Iteration}
```

zu

```
a = sqrt(a); // ITERATION
```

Entsprechend können durch den Makro

```
\sBoolValue{<Ja-Wert>}{<Nein-Wert>}
```

die Werte von `\pTrue` und `\pFalse` undefiniert werden. Somit liefern die Zeilen

```
\renewcommand{\pLanguage}{Pascal}
\sBoolValue{\textit{ja}}{\textit{nein}}
\(\pFalse = \pKey{not}\ \pTrue\)
```

das folgende Ergebnis:

```
nein = not ja
```

`\sVar` Die Makros `\sVar` und `\sKey` sind mit den Makros `\pVar` und `\pKey` iden-
`\sKey` tisch, sie werden hier nur definiert, um Kompatibilität mit früheren Versionen
`\sTrue` des `struktex.sty` zu gewährleisten. Dasselbe gilt auch für die Makros `\sTrue` und
`\sFalse` `\sFalse`.

3.3 Die Makros zur Erzeugung von Struktogrammen

`struktogramm` Die Umgebung

```
\sProofOn
\sProofOff
\PositionNSS
\begin{struktogramm}(<Breite>,<Höhe>) [<Überschrift>]
...
\end{struktogramm}
```

erzeugt Raum für ein neues Struktogramm. Die beiden Parameter legen die Breite und die Höhe des Platzes fest, der für das Struktogramm reserviert wird. Die Angaben werden in Millimetern gemacht, wobei der aktuelle Wert von `\unitlength` keine Rolle spielt. Dabei entspricht die Breite der tatsächlichen Breite, die tatsächliche Höhe wird den Erfordernissen angepasst. Stimmt die angegebene Höhe nicht mit den tatsächlichen Erfordernissen überein, läuft das Struktogramm in den umgebenden Text hinein oder es bleibt Raum frei. Es gibt einen Schalter `\sProofOn`, mit dem die angegebene Größe des Struktogramms durch vier Punkte gezeigt wird, um Korrekturen einfacher durchführen zu können. `\sProofOff` schaltet diese Hilfe entsprechend wieder ab. Die Überschrift dient zur Identifizierung des Struktogramms, wenn man sich von anderer Stelle, etwa aus einem anderen Struktogramm heraus auf dieses hier beziehen will.

Die Struktogramm-Umgebung basiert auf der `picture`-Umgebung von `LATEX`. Die in der `picture`-Umgebung gebräuchliche Längeneinheit `\unitlength` wird bei den Struktogrammen nicht benutzt; die Längeneinheit ist aus technischen Gründen auf 1 mm festgelegt. Weiterhin müssen alle Längenangaben ganzzahlige Werte sein. `\unitlength` hat zwar nach dem Zeichnen eines Struktogramms mit `StrukTEX` den gleichen Wert wie vorher, ist aber innerhalb eines Struktogramms undefiniert und darf dort auch nicht geändert werden.

`\assign` Das Hauptelement eines Struktogramms ist ein Kasten, in dem eine Operation beschrieben wird. Ein solcher Kasten wird mit `\assign` erzeugt. Die Syntax ist

Quadratwurzel von π berechnen und ausgeben

`declaration` Die `declaration`-Umgebung dient der Beschreibung von Variablen bzw. der Beschreibung der Schnittstelle. Ihre Syntax ist

```
\begin{declaration}[\langle Überschrift \rangle]
...
\end{declaration}
```

`\declarationtitle` Die Überschriftsangabe ist optional. Lässt man die Angabe weg, so wird standardmäßig die Überschrift: „Speicher bereitstellen.“ erzeugt. Will man einen anderen Text haben, wird dieser mit `\declarationtitle{\langle Überschrift \rangle}` global festgelegt. Will man für ein einzelnes Struktogramm einen speziellen Titel erzeugen, so gibt man diesen in den eckigen Klammern an.

`\description` Innerhalb der `declaration`-Umgebung werden die Beschreibungen der einzelnen Variablen mit

```
\description{\langle Variablenname \rangle}{\langle Variablenbeschreibung \rangle}
```

erzeugt. Dabei ist zu beachten, dass `\langle Variablenname \rangle` keine schließende eckige Klammer „]“ beinhalten darf, da dieser Makro mittels des `\item`-Makros definiert worden ist. Eckige Klammern sind in diesem Fall als `\lbracket` und `\rbracket` einzugeben.

Das Aussehen einer Beschreibung lässt sich mit drei Parametern steuern: `\descriptionindent`, `\descriptionwidth` und `\descriptionsep`; die Bedeutung der Parameter ist der Abbildung 1 zu entnehmen (`\xsize@nss` und `\xin@nss` sind interne Größen, die von `StüTkTEX` vorgegeben werden). Die Vorbesetzung dieser Werte ist folgendermaßen:

```
\descriptionindent=1.5em
\descriptionwidth=40pt
\descriptionsep=\tabcolsep
```

Die Bedeutung von `\descriptionwidth` ist darin zu sehen, dass ein Variablenname, der kürzer als `\descriptionwidth` ist, eine Beschreibung erhält, die auf der gleichen Höhe liegt; ansonsten wird die Beschreibung eine Zeile tiefer begonnen.

Beispiel 3

Zunächst wird nur eine einzelne Variable beschrieben.

```
\begin{struktogramm}(95,20)
  \assign%
  {%
    \begin{declaration}
      \description{\pVar{iVar}}{eine \pKey{int}-Variable,
        deren Beschreibung hier allein dem
        Zweck dient, den Makro vorzuf"uhren}
    \end{declaration}
  }
\end{struktogramm}
```

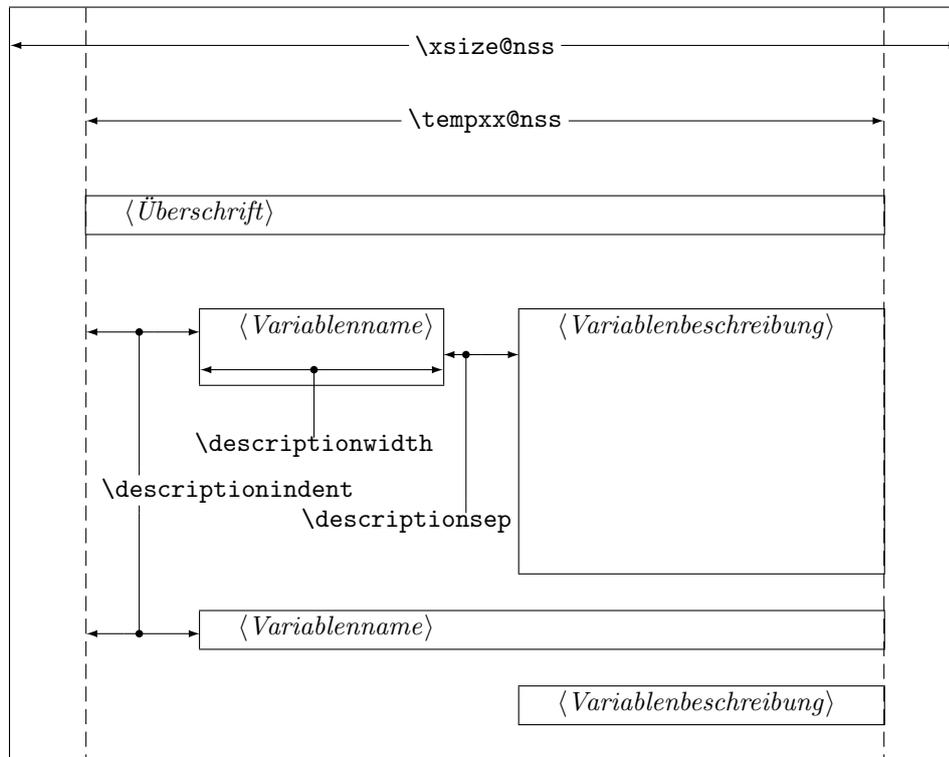


Abbildung 1: Aufbau einer Variablenbeschreibung

Das zugehörige Struktogramm, wobei zu beachten ist, dass durch die leeren eckigen Klammern keine Überschrift erzeugt wird.

Speicherplatz bereitstellen:	
iVar	{eine int-Variable, deren Beschreibung hier allein dem Zweck dient, den Makro vorzuführen}

Nun werden Variablen genauer spezifiziert:

```

\begin{struktogramm}(95,50)
  \assign{
    \begin{declaration}[Parameter:]
      \description{\pVar{iPar}}{ein \pKey{int}-Parameter,
        dessen Bedeutung hier beschrieben wird}
    \end{declaration}
    \begin{declaration}[lokale Variablen:]
      \description{\pVar{iVar}}{eine \pKey{int}-Variable,
        deren Bedeutung hier beschrieben wird}
      \description{\pVar{dVar}}{eine \pKey{double}-Variable,
        deren Bedeutung hier beschrieben wird}
    \end{declaration}
  }
\end{struktogramm}

```

Das ergibt:

Parameter:	
iPar	{ein int-Parameter, dessen Bedeutung hier beschrieben wird}
lokale Variablen:	
iVar	{eine int-Variable, deren Bedeutung hier beschrieben wird}
dVar	{eine double-Variable, deren Bedeutung hier beschrieben wird}

Zuletzt die globale Vereinbarung eines Titels:

```
\def\declarationtitle{globale Variablen:}
\begin{struktogramm}(95,13)
  {\catcode'\_ =12%
   \assign{%
     \begin{declaration}
       \description{\pVar{iVar_g}}{eine \pKey{int}-Variable}
     \end{declaration}
   }
 }
\end{struktogramm}
```

Dies ergibt das folgende Aussehen:

globale Variablen:	
iVar_g	{eine int-Variable}

Hier ist die lokale Umsetzung des `\catcodes` des Unterstrichs zu beachten, die erforderlich ist, wenn man einen Unterstrich in einem Makroargument einsetzen möchte. Diese lokale Umsetzung wird zwar auch schon bei `\pVar` gemacht, reicht aber bei der Makroexpansionstechnik von `TEX` nicht aus.

`\sub` Die Sinnbilder für einen Unterprogrammsprung und einen Aussprung aus dem
`\return` Programm sehen ähnlich aus und werden mit folgenden Befehlen gezeichnet:

```
\sub[⟨Höhe⟩]{⟨Text⟩}
\return[⟨Höhe⟩]{⟨Text⟩}
```

Die Parameter haben dieselbe Bedeutung wie bei `\assign`. Das nächste Beispiel zeigt, wie diese Sinnbilder gezeichnet werden.

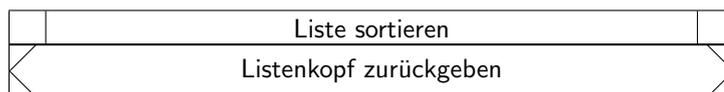
Beispiel 4

```

\begin{struktogramm}(95,20)
  \sub{Liste sortieren}
  \return{Listenkopf zur"uckgeben}
\end{struktogramm}

```

Diese Anweisungen führen zu folgendem Struktogramm:



`\while` Zum Darstellen von Schleifenkonstrukten stehen die drei Befehle `\while`, `\whileend`, `\until` und `\forever` zur Verfügung. Die While-Schleife stellt eine Wiederholung mit vorausgehender Bedingungsprüfung (kopfgesteuerte Schleife) dar, die `\until`-Schleife testet die Bedingung am Schleifenende (fußgesteuerte Schleife) und die `\forever`-Schleife ist eine Endlosschleife, aus der man mit dem Befehl `\exit` herauspringen kann.

```

\while[⟨Breite⟩]{⟨Text⟩}⟨Unterstruktogramm⟩\whileend
\until[⟨Breite⟩]{⟨Text⟩}⟨Unterstruktogramm⟩\untilend
\forever[⟨Breite⟩]⟨Unterstruktogramm⟩\foreverend
\exit[⟨Höhe⟩]{⟨Text⟩}

```

$\langle Breite \rangle$ ist die Dicke des Rahmens des Sinnbildes, $\langle Text \rangle$ ist der Bedingungstext, der in diesen Rahmen geschrieben wird. Wird die Breite nicht angegeben, richtet sich die Rahmendicke nach der Höhe des Textes. Der Text wird linksbündig in den Rahmen geschrieben. Ist der Text leer, so wird ein dünner Rahmen gezeichnet

An Stelle von $\langle Unterstruktogramm \rangle$ können beliebige Befehle von \TeX stehen (mit Ausnahme von `\openstrukt` und `\closestrukt`), die das Struktogramm innerhalb der `\while`-, der `\until`- oder der `\forever`-Schleife bilden.

Um Kompatibilität mit der Weiterentwicklung des `struktex.sty` von J. Dietel zu erreichen, gibt es die Makros `\dfr` und `\dfrend` mit derselben Bedeutung wie `\forever` und `\foreverend`.

Die folgenden Beispiele zeigen den Einsatz der `\while`- und `\until`-Makros, `\forever` wird weiter unten gezeigt.

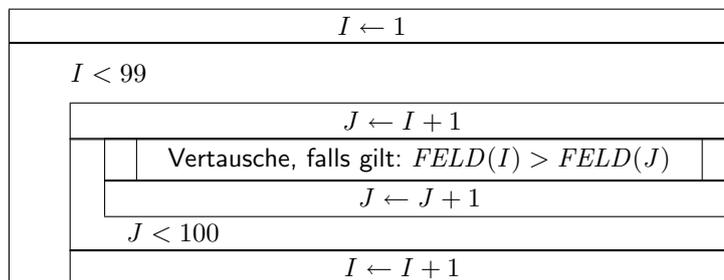
Beispiel 5

```

\begin{struktogramm}(95,40)
  \assign{\(I \gets 1\)}
  \while[8]{\(I < 99\)}
    \assign{\(J \gets I+1\)}
    \until{\(J < 100\)}
      \sub{Vertausche, falls gilt: \(\text{FELD}(I) > \text{FELD}(J)\)}
      \assign{\(J \gets J+1\)}
    \untilend
  \assign{\(I \gets I+1\)}
\whileend
\end{struktogramm}

```

Diese Anweisungen führen zu folgendem Struktogramm:



Die `\exit`-Anweisung gibt nur im Zusammenhang mit einfachen oder mehrfachen Verzweigungen Sinn, daher wird sie im Anschluss an die Diskussion der Verzweigungen diskutiert.

`\ifthenelse` Zur Darstellung von Alternativen stellt `StylTeX` die Sinnbilder für einen If-Then-Else-Block und für mehrfache Alternativen eine Case-Konstruktion zur Verfügung. Da in der `Picture`-Umgebung von `LATEX` nur Linien mit bestimmten Steigungen gezeichnet werden können, muss der Benutzer bei beiden Befehlen selbst den Winkel bestimmen, mit dem die notwendigen schrägen Linien gezeichnet werden sollen (hier ist also etwas mehr ‚Handarbeit‘ nötig).

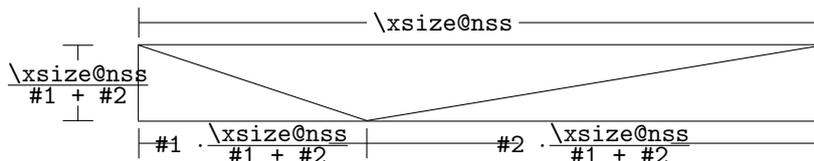
Wenn hingegen der `curves.sty` bzw. der `emlines2.sty` eingesetzt wird, ist die Darstellung von Geraden mit beliebiger Steigung möglich.

Der If-Then-Else-Befehl sieht so aus:

```

\ifthenelse[⟨Höhe⟩]{⟨Linker Winkel⟩}{⟨Rechter Winkel⟩}
  {⟨Bedingung⟩}{⟨Linker Text⟩}{⟨Rechter Text⟩}
  ⟨Unterstruktogramm⟩
\change
  ⟨Unterstruktogramm⟩
\ifend
  
```

Für den Fall, dass das optionale Argument `⟨Höhe⟩` nicht angegeben ist, sind `⟨Linker Winkel⟩` (`#1`) und `⟨Rechter Winkel⟩` (`#2`) Ziffern zwischen 1 und 6; diese bestimmen die Steigung der beiden Unterteilungslinien des If-Then-Else-Blocks (großer Wert = kleine Steigung). Größere Werte werden auf 6 gesetzt, kleinere auf 1. Das genaue Verhalten der Steigungen ist dem folgenden Bild zu entnehmen; `\xsize@nss` ist dabei die Breite des aktuellen Unterstruktogrammes. Wird die `⟨Höhe⟩` vorgegeben, so bestimmt dieser Wert statt des Ausdruckes $\frac{\text{xsize@nss}}{\#1 + \#2}$ die Höhe des Bedingungsrechteckes.



`⟨Bedingung⟩` wird in das so gebildete obere mittlere Dreieck gesetzt; die Parameter `⟨Linker Text⟩` und `⟨Rechter Text⟩` werden in das linke bzw. rechte untere Dreieck gesetzt. Der Bedingungs-Text kann in seinem Dreiecks-Feld umgebrochen werden.

Ab Version 5.3 wird der Bedingungstext durch geeigneten Umbruch beliebigen Steigungen angepasst.³ Die beiden anderen Texte sollten kurz sein (z. B. ja/nein oder true/false), da sie nicht umgebrochen werden können und sonst über ihr Dreiecks-Feld hinausragen. Um an dieser Stelle Einheitlichkeit zu erzielen, sollten die Makros `\pTrue` und `\pFalse` benutzt werden. Hinter `\ifthenelse` werden die Befehle für das linke, hinter `\change` die für das rechte „Unterstruktogramm“ geschrieben. Falls diese beiden Struktogramme nicht gleich lang sind, wird ein Kasten mit einem \emptyset ergänzt.⁴ Mit `\ifend` wird das If-Then-Else-Element beendet. Es folgen zwei Beispiele für die Anwendung.

Beispiel 6

```
\begin{struktogramm}(95,32)
  \ifthenelse[12]{1}{2}
    {Flag f"ur Drucker-Ausgabe gesetzt ?}{\sTrue}{\sFalse}
    \assign[15]{Ausgabe auf Drucker umleiten}
  \change
    \assign{Ausgabe auf den Bildschirm}
  \ifend
\end{struktogramm}
```

Diese Anweisungen führen zu folgendem Struktogramm:

Flag für Drucker-Ausgabe gesetzt ?	
WAHR	FALSCH
Ausgabe auf Drucker umleiten	Ausgabe auf den Bildschirm
	\emptyset

Beispiel 7

```
\begin{struktogramm}(90,30)
  \ifthenelse{3}{4}
    {Flag f"ur Drucker-Ausgabe gesetzt ?}{\sTrue}{\sFalse}
    \assign[15]{Ausgabe auf Drucker umleiten}
  \change
    \assign{Ausgabe auf den Bildschirm}
  \ifend
\end{struktogramm}
```

Diese Anweisungen führen zu folgendem Struktogramm:

³Diese Erweiterung stammt von Daniel Hagedorn, dem ich hiermit herzlich danken möchte

⁴Eventuell ist ein `\strut` hilfreich, um unterschiedliche Höhen von Texten auszugleichen.

Flag für Drucker-Ausgabe gesetzt ?	
WAHR	FALSCH
Ausgabe auf Drucker umleiten	Ausgabe auf den Bildschirm
	∅

`\case`
`\switch`
`\caseend`

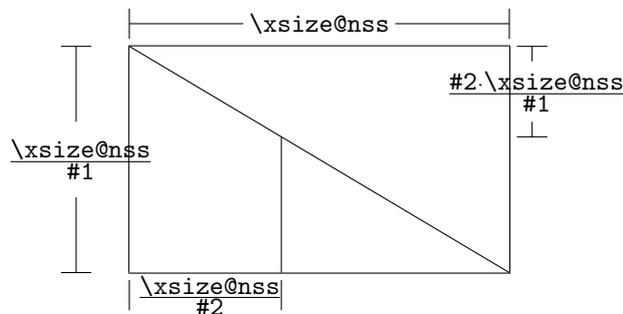
Das Case-Konstrukt hat folgende Syntax:

```

\case[⟨Höhe⟩]{⟨Winkel⟩}{⟨Anzahl der Fälle⟩}{⟨Bedingung⟩}{⟨Text
des 1. Falles⟩}
  ⟨Unterstruktogramm⟩
\switch[⟨Position⟩]{⟨Text des 2. Falles⟩}
  ⟨Unterstruktogramm⟩
...
\switch[⟨Position⟩]{⟨Text des n. Falles⟩}
  ⟨Unterstruktogramm⟩
\caseend

```

Ist die $\langle \text{Höhe} \rangle$ nicht angegeben, so erhält die Unterteilungslinie des Case-Sinnbildes die durch $\langle \text{Winkel} \rangle$ angegebene Steigung (die bei `\ifthenelse` erwähnten Winkelwerte). In das obere der durch diese Linie entstandenen beiden Dreieck wird der Text $\langle \text{Bedingung} \rangle$ gesetzt. Die Größenverhältnisse ergeben sich aus der folgenden Skizze (`\xsize@nss` ist die aktuelle Breite des (Unter-)Struktogramms):



Der zweite Parameter $\langle \text{Anzahl der Fälle} \rangle$ gibt die Anzahl der zu zeichnenden Fälle an; alle Unterstruktogramme der einzelnen Fälle erhalten die gleiche Breite. Der $\langle \text{Text des 1. Falles} \rangle$ muss als Parameter des `\case`-Befehls angegeben werden, alle weiteren Fälle werden über den `\switch`-Befehl eingeleitet. Hinter dem Text folgen dann die Befehle für das eigentliche Unterstruktogramm des jeweiligen Falles. Der letzte Fall wird mit `\caseend` abgeschlossen. Ein Case-Sinnbild mit drei Fällen zeigt das folgende Beispiel.

Beispiel 8

```

\begin{struktogramm}(95,30)
\case{4}{3}{Signum(x)}{-1}
\assign{z \gets - \frac{1}{x}}

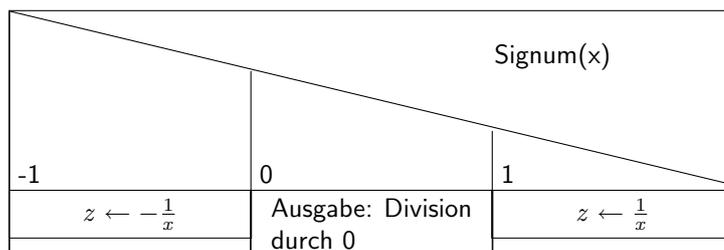
```

```

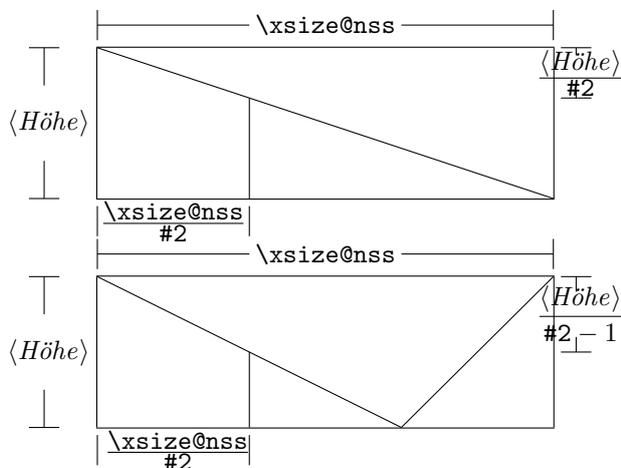
\switch{0}
  \assign{Ausgabe: Division durch 0}
\switch{1}
  \assign{$z \gets \frac{1}{x}$}
\caseend
\end{struktogramm}

```

Diese Anweisungen führen zu folgendem Struktogramm:



Der optionale Parameter [$\langle Höhe \rangle$] ist nur einzusetzen, wenn die Option „curves“, „emlines2“ oder „pict2e“ gesetzt ist; ist das nicht der Fall, können die Struktogramme durcheinander kommen. Die Erweiterung des `\switch`-Kommandos mit [$\langle Höhe \rangle$] führt zu einer anderen Bedeutung von $\langle Winkel \rangle$. Ist der Wert gerade, wird wie zuvor eine gerade Linie zur Aufteilung des zugrundeliegenden Rechtecks gezeichnet; ist der Wert hingegen ungerade, wird der letzte Fall wie im folgenden gezeigt als Sonderfall gezeichnet.



Beispiel 9

```

\begin{struktogramm}(95,30)
  \case[10]{4}{3}{Signum(x)}{-1}
    \assign{$z \gets - \frac{1}{x}$}
  \switch{0}
    \assign{Ausgabe: Division durch 0}
  \switch{1}
    \assign{$z \gets \frac{1}{x}$}
\end{struktogramm}

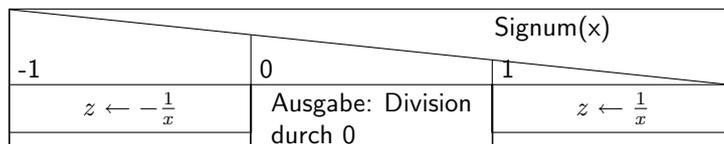
```

```

\caseend
\end{struktogramm}

```

Diese Anweisungen führen zu folgendem Struktogramm:



Ist der erste Parameter hingegen ungerade, wird ein Standardzweig gezeichnet; der Wert für den Standardfall sollte dann rechtsbündig ausgerichtet werden.

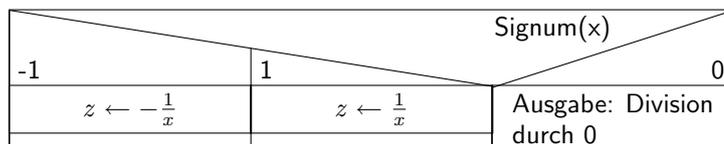
Beispiel 10

```

\begin{struktogramm}(95,30)
\case[10]{5}{3}{Signum(x)}{-1}
\assign{$z \gets -\frac{1}{x}$}
\switch{1}
\assign{$z \gets \frac{1}{x}$}
\switch[r]{0}
\assign{Ausgabe: Division durch 0}
\caseend
\end{struktogramm}

```

Diese Anweisungen führen zu folgendem Struktogramm:



Das folgende Beispiel zeigt, wie mittels einfacher Verzweigung aus einer Endlosschleife gesprungen werden kann. Das Beispiel lässt sich ohne weiteres auf eine mehrfache Verzweigung übertragen.

Beispiel 11

```

\begin{struktogramm}(95,40)
\forever
\assign{lies Zeichen}
\ifthenelse{3}{3}{Zeichen = 'E'?}
{j}{n}
\exit{Sprung hinter Schleife}
\change
\ifend

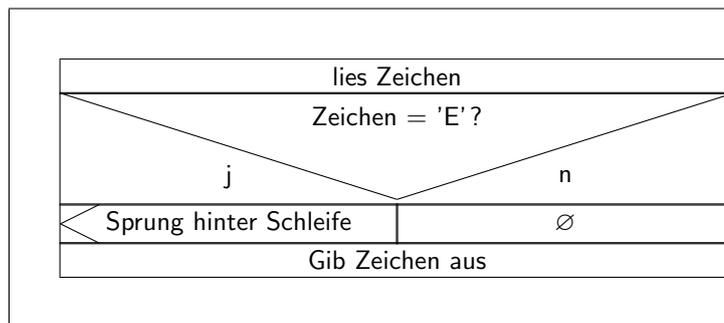
```

```

\assign{Gib Zeichen aus}
\foreverend
\end{struktogramm}

```

Diese Anweisungen führen zu folgendem Struktogramm:



`centernss` Soll ein Struktogramm zentriert dargestellt werden, so wird dazu die Umgebung

```

\begin{centernss}
  <Struktogramm>
\end{centernss}

```

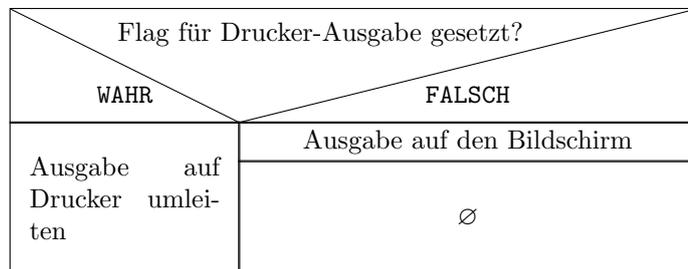
benutzt:

```

\begin{centernss}
\begin{struktogramm}(90,35)
  \ifthenelse{2}{4}
    {Flag f"ur Drucker-Ausgabe gesetzt?}{\sTrue}{\sFalse}%
    \assign[20]{Ausgabe auf Drucker umleiten}
  \change
    \assign{Ausgabe auf den Bildschirm}
  \ifend
\end{struktogramm}
\end{centernss}

```

Das führt zu folgendem:



`\CenterNssFile` Häufig gibt es den Fall, dass Struktogramme in eigenen Dateien abgelegt werden, damit sie für sich allein auf Korrektheit getestet werden können oder in anderen Zusammenhängen genutzt werden können. Sollen sie zentriert eingebunden werden, kann nicht mit der folgenden Konstruktion gearbeitet werden:

```

\begin{center}
  \input{...}
\end{center}

```

da auf diese Weise der gesamte Text innerhalb des Struktogramms zentriert würde. Um diesen Fall einfach und korrekt abhandeln zu können, kann das Makro `\CenterNssFile` eingesetzt werden, das auch in der Schreibweise `centernssfile` definiert ist. Voraussetzung ist, dass die Datei, die die Anweisungen für das Struktogramm enthält, die Dateinamenserweiterung `.nss` hat, der Name der einzubindenden Datei *muss* demzufolge ohne Erweiterung angegeben werden. Wenn die Datei `struktex-test-0.nss` das in Abschnitt 4, Zeile 2–10 gezeigte Aussehen hat, so führt die Anweisung

```
\centernssfile{struktex-test-0}
```

zu folgendem Aussehen des formatierten Textes:

Text		
-1	0	1
$z \leftarrow -\frac{1}{x}$	Ausgabe: Division durch 0	$z \leftarrow \frac{1}{x}$

`\openstrukt` Diese beiden Makros sind nur der Kompatibilität zu vorherigen Versionen
`\closestrukt` von \LaTeX willen noch erhalten. Von der Bedeutung her entsprechen sie
`\struktogramm` und `\endstruktogramm`. Die Syntax ist

```
\openstrukt[width][height]
```

und

```
\closestrukt.
```

`\assert` Der Makro `\assert` wurde eingeführt, um die Verifikation von Algorithmen zu unterstützen, er ist aber nur aktiv, wenn die Stil-Option `verification` gesetzt wurde. Er dient dazu, an ausgewählten Stellen Zusicherungen über den Zustand von Variablen zu markieren, die Syntax entspricht dem `\assign`:

```
\assert[Höhe]{Zusicherung}
```

Sein Einsatz ergibt sich aus dem folgenden:

```

\begin{struktogramm}(70,20)[Zusicherungen in Struktogrammen]
  \assign{\(a\gets a^2\)}
  \assert{\(a\ge 0\)}
\end{struktogramm}

```

Das dazugehörige Struktogramm sieht folgendermaßen aus:

Zusicherungen in Struktogrammen

$a \leftarrow a^2$
$a \geq 0$

4 Beispieldatei zum Einbinden in die Dokumentation

Die folgenden Zeilen bilden eine Beispieldatei, die bei der Erstellung dieser Dokumentation benötigt wird.

```
40 \begin{struktogramm}(95,40) [Text]
41   \case[10]{3}{3}{Signum(x)}{-1}
42     \assign{(z \gets - \frac{1}{x}\)}
43   \switch{0}
44     \assign{Ausgabe: Division durch 0}
45   \switch[r]{1}
46     \assign{(z \gets \frac{1}{x}\)} \caseend
47 \end{struktogramm}
```

5 Verschiedene Beispieldateien

5.1 Beispieldatei zum Austesten der Makros des `struktex.sty` ohne die Benutzung optionaler Pakete

Die folgenden Zeilen bilden eine Musterdatei, die zum Austesten der Makros benutzt werden kann. Der Inhalt ist nur in Englisch vorhanden.

```
48 \documentclass[draft]{article}
49 \usepackage{struktex}
50
51 \begin{document}
52
53 \begin{struktogramm}(90,137)
54   \assign%
55   {
56     \begin{declaration}[]
57       \description{(a, b, c\)}{three variables which are to be sorted}
58       \description{(tmp\)}{temporary variable for the circular swap}
59     \end{declaration}
60   }
61   \ifthenelse{1}{2}{(a\le c\)}{j}{n}
62   \change
63   \assign{(tmp\gets a\)}
64   \assign{(a\gets c\)}
65   \assign{(c\gets tmp\)}
66   \ifend
67   \ifthenelse{2}{1}{(a\le b\)}{j}{n}
68   \ifthenelse{1}{1}{(b\le c\)}{j}{n}
69   \change
70   \assign{(tmp\gets c\)}
```

```

71 \assign{\(c\gets b\)}
72 \assign{\(b\gets tmp\)}
73 \ifend
74 \change
75 \assign{\(tmp\gets a\)}
76 \assign{\(a\gets b\)}
77 \assign{\(b\gets tmp\)}
78 \ifend
79 \end{struktogramm}
80
81 \end{document}

```

5.2 Beispieldatei zum Austesten der Makros des `struktex.sty` mit dem Paket `pict2e.sty`

Die folgenden Zeilen bilden eine Musterdatei, die zum Austesten der Makros benutzt werden kann. Der Inhalt ist nur in Englisch vorhanden.

```

82 \documentclass{article}
83 \usepackage[pict2e, verification]{struktex}
84
85 \begin{document}
86 \def\StruktBoxHeight{7}
87 %\sProofOn{}
88 \begin{struktogramm}(90,137)
89   \assign%
90   {
91     \begin{declaration}[]
92       \description{\(a, b, c\)}{three variables which are to be sorted}
93       \description{\(tmp\)}{temporary variable for the circular swap}
94     \end{declaration}
95   }
96   \assert[\StruktBoxHeight]{\sTrue}
97   \ifthenelse[\StruktBoxHeight]{1}{2}{\(\(a\le c\))}{j}{n}
98     \assert[\StruktBoxHeight]{\(\(a\le c\))}
99   \change
100     \assert[\StruktBoxHeight]{\(\(a>c\))}
101     \assign[\StruktBoxHeight]{\(\(tmp\gets a\))}
102     \assign[\StruktBoxHeight]{\(\(a\gets c\))}
103     \assign[\StruktBoxHeight]{\(\(c\gets tmp\))}
104     \assert[\StruktBoxHeight]{\(\(a<c\))}
105   \ifend
106   \assert[\StruktBoxHeight]{\(\(a\le c\))}
107   \ifthenelse[\StruktBoxHeight]{2}{1}{\(\(a\le b\))}{j}{n}
108     \assert[\StruktBoxHeight]{\(\(a\le b \wedge a\le c\))}
109     \ifthenelse[\StruktBoxHeight]{1}{1}{\(\(b\le c\))}{j}{n}
110       \assert[\StruktBoxHeight]{\(\(a\le b \le c\))}
111     \change
112       \assert[\StruktBoxHeight]{\(\(a \le c < b\))}
113       \assign[\StruktBoxHeight]{\(\(tmp\gets c\))}
114       \assign[\StruktBoxHeight]{\(\(c\gets b\))}
115       \assign[\StruktBoxHeight]{\(\(b\gets tmp\))}
116       \assert[\StruktBoxHeight]{\(\(a\le b < c\))}
117     \ifend

```

```

118 \change
119 \assert[\StruktBoxHeight]{\ (b < a\le c\)}
120 \assign[\StruktBoxHeight]{\ (tmp\gets a\)}
121 \assign[\StruktBoxHeight]{\ (a\gets b\)}
122 \assign[\StruktBoxHeight]{\ (b\gets tmp\)}
123 \assert[\StruktBoxHeight]{\ (a<b\le c\)}
124 \ifend
125 \assert[\StruktBoxHeight]{\ (a\le b \le c\)}
126 \end{struktoqramm}
127
128 \end{document}

```

5.3 Beispieldatei zum Austesten der Makros des `struktxp.sty`

Die folgenden Zeilen bilden eine Musterdatei, die zum Austesten der Makros des `struktxp.sty` benutzt werden kann. Zum Testen sollten auch die Kommentarzeichen vor der Zeile `\usepackage[T1]{fontenc}` gelöscht werden. Der Text ist nur in Englisch vorgegeben.

```

129 \documentclass{article}
130
131 \usepackage{struktxp,struktxf}
132
133 \nofiles
134
135 \begin{document}
136
137 \pLanguage{Pascal}
138 \section*{Default values (Pascal):}
139
140 {\obeylines
141 Variables: \pVar{iV_g}, \sVar{zV^}, \pVariable{&i}
142 Keywords: \pKeyword{begin}, \pKey{while}, \sKey{__CPP__}
143 in math mode: \(\pVar{a}+\pVar{iV_g}\)
144 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
145 }
146
147 \paragraph{After changing the boolean values with}
148 \verb-\pBoolValue{yes}{no}-:
149
150 {\obeylines
151 \pBoolValue{yes}{no}
152 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
153 }
154
155 \paragraph{after changing the fonts with}
156 \verb-\pFonts{\itshape}{\sffamily\bfseries}{}-:
157
158 {\obeylines
159 \pFonts{\itshape}{\sffamily\bfseries}{}
160 Variables: \pVar{iV_g}, \sVar{zV^}, \pVariable{&i}
161 Keywords: \pKeyword{begin}, \pKey{while}, \sKey{__CPP__}
162 in math mode: \(\pVar{a}+\pVar{iV_g}\)
163 boolean values: \sTrue, \sFalse, \pTrue, \pFalse

```

```

164 }
165
166 \paragraph{after changing the fonts with}
167 \verb-\pFonts{\ttfamily}{\ttfamily\bfseries}{\ttfamily\slshape}-:
168
169 {\obeylines
170 \pFonts{\ttfamily}{\ttfamily\bfseries}{\ttfamily\slshape}
171 Variables: \pVar{iV_g}, \sVar{zV^}, \pVariable{&i}
172 Keywords: \pKeyword{begin}, \pKey{while}, \sKey{__CPP__}
173 in math mode: \(\pVar{a}+\pVar{iV_g}\)
174 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
175 }
176
177 \paragraph{after changing the fonts with}
178 \verb-\pFonts{\itshape}{\bfseries\itshape}{}-:
179
180 {\obeylines
181 \pFonts{\itshape}{\bfseries\itshape}{}
182 Variables: \pVar{iV_g}, \sVar{zV^}, \pVariable{&i}
183 Keywords: \pKeyword{begin}, \pKey{while}, \sKey{__CPP__}
184 in math mode: \(\pVar{a}+\pVar{iV_g}\)
185 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
186
187 \vspace{15pt}
188 Without \textit{italic correction}:
189 M \pVar{M} M \pKey{M} M. \pVar{M}. M. \pKey{M}. M.
190 }
191
192 \pLanguage{C}
193 \pBoolValue{\texttt{WAHR}}{\texttt{FALSCH}}
194 \section*{Default values (C):}
195
196 {\obeylines
197 Variables: \pVar{iV_g}, \sVar{zV^}, \pVariable{&i}
198 Keywords: \pKeyword{begin}, \pKey{while}, \sKey{__CPP__}
199 in math mode: \(\pVar{a}+\pVar{iV_g}\)
200 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
201 }
202
203 \paragraph{After changing the boolean values with}
204 \verb-\pBoolValue{\texttt{yes}}{\texttt{no}}-:
205
206 {\obeylines
207 \pBoolValue{\texttt{yes}}{\texttt{no}}
208 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
209 }
210
211 \paragraph{after changing the fonts with}
212 \verb-\pFonts{\itshape}{\sffamily\bfseries}{}-:
213
214 {\obeylines
215 \pFonts{\itshape}{\sffamily\bfseries}{}
216 Variables: \pVar{iV_g}, \sVar{zV^}, \pVariable{&i}
217 Keywords: \pKeyword{begin}, \pKey{while}, \sKey{__CPP__}

```

```

218 in math mode: \(\pVar{a}+\pVar{iV_g}\)
219 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
220 }
221
222 \paragraph{after changing the fonts with}
223 \verb-\pFonts{\ttfamily}{\ttfamily\bfseries}{\ttfamily\slshape}-:
224
225 {\obeylines
226 \pFonts{\ttfamily}{\ttfamily\bfseries}{\ttfamily\slshape}
227 Variables: \pVar{iV_g}, \sVar{zV^}, \pVariable{&i}
228 Keywords: \pKeyword{begin}, \pKey{while}, \sKey{__CPP__}
229 in math mode: \(\pVar{a}+\pVar{iV_g}\)
230 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
231 }
232
233 \paragraph{after changing the fonts with}
234 \verb-\pFonts{\itshape}{\bfseries\itshape}{}-:
235
236 {\obeylines
237 \pFonts{\itshape}{\bfseries\itshape}{}
238 Variables: \pVar{iV_g}, \sVar{zV^}, \pVariable{&i}
239 Keywords: \pKeyword{begin}, \pKey{while}, \sKey{__CPP__}
240 in math mode: \(\pVar{a}+\pVar{iV_g}\)
241 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
242
243 \vspace{15pt}
244 Without \textit{italic correction}:
245     M \pVar{M} M \pKey{M} M. \pVar{M}. M. \pKey{M}. M.
246 }
247
248 \pLanguage{Java}
249 \pBoolValue{\texttt{WAHR}}{\texttt{FALSCH}}
250 \section*{Default values (Java):}
251
252 {\obeylines
253 Variables: \pVar{iV_g}, \sVar{zV^}, \pVariable{&i}
254 Keywords: \pKeyword{begin}, \pKey{while}, \sKey{__CPP__}
255 in math mode: \(\pVar{a}+\pVar{iV_g}\)
256 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
257 }
258
259 \paragraph{After changing the boolean values with}
260 \verb-\pBoolValue{\texttt{yes}}{\texttt{no}}-:
261
262 {\obeylines
263 \pBoolValue{\texttt{yes}}{\texttt{no}}
264 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
265 }
266
267 \paragraph{after changing the fonts with}
268 \verb-\pFonts{\itshape}{\sffamily\bfseries}{}-:
269
270 {\obeylines
271 \pFonts{\itshape}{\sffamily\bfseries}{}

```

```

272 Variables: \pVar{iV_g}, \sVar{zV^}, \pVariable{&i}
273 Keywords: \pKeyword{begin}, \pKey{while}, \sKey{__CPP__}
274 in math mode: \(\pVar{a}+\pVar{iV_g}\)
275 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
276 }
277
278 \paragraph{after changing the fonts with}
279 \verb-\pFonts{\ttfamily}{\ttfamily\bfseries}{\ttfamily\slshape}-:
280
281 {\obeylines
282 \pFonts{\ttfamily}{\ttfamily\bfseries}{\ttfamily\slshape}
283 Variables: \pVar{iV_g}, \sVar{zV^}, \pVariable{&i}
284 Keywords: \pKeyword{begin}, \pKey{while}, \sKey{__CPP__}
285 in math mode: \(\pVar{a}+\pVar{iV_g}\)
286 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
287 }
288
289 \paragraph{after changing the fonts with}
290 \verb-\pFonts{\itshape}{\bfseries\itshape}{}-:
291
292 {\obeylines
293 \pFonts{\itshape}{\bfseries\itshape}{}
294 Variables: \pVar{iV_g}, \sVar{zV^}, \pVariable{&i}
295 Keywords: \pKeyword{begin}, \pKey{while}, \sKey{__CPP__}
296 in math mode: \(\pVar{a}+\pVar{iV_g}\)
297 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
298
299 \vspace{15pt}
300 Without \textit{italic correction}:
301 M \pVar{M} M \pKey{M} M. \pVar{M}. M. \pKey{M}. M.
302 }
303
304 \pLanguage{Python}
305 \pBoolValue{\texttt{WAHR}}{\texttt{FALSCH}}
306 \section*{Default values (Python):}
307
308 {\obeylines
309 Variables: \pVar{iV_g}, \sVar{zV^}, \pVariable{&i}
310 Keywords: \pKeyword{begin}, \pKey{while}, \sKey{__CPP__}
311 in math mode: \(\pVar{a}+\pVar{iV_g}\)
312 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
313 }
314
315 \paragraph{After changing the boolean values with}
316 \verb-\pBoolValue{\texttt{yes}}{\texttt{no}}-:
317
318 {\obeylines
319 \pBoolValue{\texttt{yes}}{\texttt{no}}
320 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
321 }
322
323 \paragraph{after changing the fonts with}
324 \verb-\pFonts{\itshape}{\sffamily\bfseries}{}-:
325

```

```

326 {\obeylines
327 \pFonts{\itshape}{\sffamily\bfseries}{}
328 Variables: \pVar{iV_g}, \sVar{zV^}, \pVariable{&i}
329 Keywords: \pKeyword{begin}, \pKey{while}, \sKey{__CPP__}
330 in math mode: \(\pVar{a}+\pVar{iV_g}\)
331 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
332 }
333
334 \paragraph{after changing the fonts with}
335 \verb-\pFonts{\ttfamily}{\ttfamily\bfseries}{\ttfamily\slshape}-:
336
337 {\obeylines
338 \pFonts{\ttfamily}{\ttfamily\bfseries}{\ttfamily\slshape}
339 Variables: \pVar{iV_g}, \sVar{zV^}, \pVariable{&i}
340 Keywords: \pKeyword{begin}, \pKey{while}, \sKey{__CPP__}
341 in math mode: \(\pVar{a}+\pVar{iV_g}\)
342 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
343 }
344
345 \paragraph{after changing the fonts with}
346 \verb-\pFonts{\itshape}{\bfseries\itshape}{}-:
347
348 {\obeylines
349 \pFonts{\itshape}{\bfseries\itshape}{}
350 Variables: \pVar{iV_g}, \sVar{zV^}, \pVariable{&i}
351 Keywords: \pKeyword{begin}, \pKey{while}, \sKey{__CPP__}
352 in math mode: \(\pVar{a}+\pVar{iV_g}\)
353 boolean values: \sTrue, \sFalse, \pTrue, \pFalse
354
355 \vspace{15pt}
356 Without \textit{italic correction}:
357 M \pVar{M} M \pKey{M} M. \pVar{M}. M. \pKey{M}. M.
358 }
359
360 \end{document}
361 %%
362 %% End of file 'struktex-test-2.tex'.

```

5.4 Beispieldatei zum Austesten der Makros des `struktexp.sty`

Die folgenden Zeilen werden in einem anderen Zusammenhang benutzt, Java-Methoden zu dokumentieren. An dieser Stelle wird ein eigener Weg gewählt, da das sonst übliche Arbeiten mit `lstinline` zu Fehlern führt.

```

363 \documentclass{article}
364
365 \usepackage{struktexp,struktxf}
366
367 \makeatletter
368 \newlength{\fdesc@len}
369 \newcommand{\fdesc@label}[1]%
370 {%
371   \settowidth{\fdesc@len}{\fdesc@font #1}%
372   \advance\hspace by -2em

```

```

373 \ifdim\fdesc@len>\hsize%                % term > labelwidth
374   \parbox[b]{\hsize}%
375   {%
376     \fdesc@font #1%
377   }\\%
378 \else%                                    % term < labelwidth
379 \ifdim\fdesc@len>\labelwidth%            % term > labelwidth
380   \parbox[b]{\labelwidth}%
381   {%
382     \makebox[Opt][l]{\fdesc@font #1}\\%
383   }%
384 \else%                                    % term < labelwidth
385   {\fdesc@font #1}%
386 \fi\fi%
387 \hfil\relax%
388 }
389 \newenvironment{fdescription}[1][\tt]%
390 {%
391   \def\fdesc@font{#1}
392   \begin{quote}%
393   \begin{list}{}%
394   {%
395     \renewcommand{\makelabel}{\fdesc@label}%
396     \setlength{\labelwidth}{120pt}%
397     \setlength{\leftmargin}{\labelwidth}%
398     \addtolength{\leftmargin}{\labelsep}%
399   }%
400 }%
401 {%
402   \end{list}%
403   \end{quote}%
404 }
405 \makeatother
406
407 \pLanguage{Java}
408
409 \begin{document}
410
411 \begin{fdescription}
412 \item[\index{Methoden}>drawImage(Image img,
413                                     int dx1,
414                                     int dy1,
415                                     int dx2,
416                                     int dy2,
417                                     int sx1,
418                                     int sy1,
419                                     int sx2,
420                                     int sy2,
421                                     ImageObserver observer)=%
422 \Expr{\pKey{public} \pKey{abstract} \pKey{boolean} drawImage(Image img,
423                                     \pKey{int} dx1,
424                                     \pKey{int} dy1,
425                                     \pKey{int} dx2,
426                                     \pKey{int} dy2,

```

```

427             \pKey{int} sx1,
428             \pKey{int} sy1,
429             \pKey{int} sx2,
430             \pKey{int} sy2,
431             ImageObserver observer)}}%
432     \pExp{public abstract boolean drawImage(Image img, int dx1, int
433         dy1, int dx2, int dy2, int sx1, int sy1, int sx2, int sy2,
434         ImageObserver observer)}}%
435 \ldots
436 \end{fdescription}
437 \end{document}
438 %%
439 %% End of file 'struktex-test-5.tex'.

```

6 Makros zur Erstellung der Dokumentation des `struktex.sty`

Um die Formatierung der Dokumentation ein wenig zu erleichtern, werden ein paar Makros benutzt, die in einer eigenen `.sty`-Datei zusammengefasst wurden. Ein wesentlicher Teil beruht auf einer Modifikation der `newtheorem`-Umgebung aus `latex.sty` zur Auszeichnung der Beispiele, die Implementation der Abkürzungen wurde in [\[Neu96\]](#) vorgeschlagen.

Dazu wurden aus dem `verbatim.sty` einige Kommandos übernommen und modifiziert, damit das Schreiben und Lesen von Dateien im `verbatim`-Modus auch im Zusammenhang mit dem `docstrip`-Paket funktioniert. Schließlich wurde auch noch eine Idee von Tobias Oetiker aus seinem `layout.sty`, der im Zusammenhang mit *lshort2e.tex - The not so short introduction to LaTeX2e* entstand, zum parallelen Setzen von Quelle und formatiertem Text genutzt.

```

440 (*strukdoc)
441 \RequirePackage{ifpdf}
442 \ProvidesPackage{strukdoc}
443     [\filedate\space\fileversion\space (Jobst Hoffmann)]
444 \newif\ifcolor \IfFileExists{color.sty}{\colortrue}{}
445 \ifpdf \RequirePackage[colorlinks]{hyperref}\else
446     \def\href#1{\texttt}\fi
447 \ifcolor \RequirePackage{color}\fi
448 \RequirePackage{nameref}
449 \RequirePackage{url}
450 \renewcommand\ref{\protect\T@ref}
451 \renewcommand\pageref{\protect\T@pageref}
452 \@ifundefined{zB}{}{\endinput}
453 \providecommand\pparg[2]{%
454     {\ttfamily}\meta{#1},\meta{#2}{\ttfamily}}
455 \providecommand\envb[1]{%
456     {\ttfamily}\char'\begin\char'\{#1\char'\}}
457 \providecommand\enve[1]{%
458     {\ttfamily}\char'\end\char'\{#1\char'\}}
459 \newcommand{\zBspace}{z.,B.}
460 \let\zB=\zBspace
461 \newcommand{\dhspace}{d.,h.}
462 \let\dh=\dhspace

```

```

463 \let\foreign=\textit
464 \newcommand\Abb[1]{Abbildung~\ref{#1}}
465 \def\newexample#1{%
466   \@ifnextchar[{\@oexmpl{#1}}{\@nexmpl{#1}}
467 \def\@nexmpl#1#2{%
468   \@ifnextchar[{\@xnexmpl{#1}{#2}}{\@ynexmpl{#1}{#2}}
469 \def\@xnexmpl#1#2[#3]{%
470   \expandafter\@ifdefinable\csname #1\endcsname
471     {\@definecounter{#1}\@newctr{#1}[#3]}%
472   \expandafter\xdef\csname the#1\endcsname{%
473     \expandafter\noexpand\csname the#3\endcsname \@exmplcountersep
474     \@exmplcounter{#1}}%
475   \global\@namedef{#1}{\@exmpl{#1}{#2}}%
476   \global\@namedef{end#1}{\@endexample}}
477 \def\@ynexmpl#1#2{%
478   \expandafter\@ifdefinable\csname #1\endcsname
479     {\@definecounter{#1}}%
480   \expandafter\xdef\csname the#1\endcsname{\@exmplcounter{#1}}%
481   \global\@namedef{#1}{\@exmpl{#1}{#2}}%
482   \global\@namedef{end#1}{\@endexample}}
483 \def\@oexmpl#1[#2]#3{%
484   \@ifundefined{c@#2}{\@nocounterr{#2}}%
485   {\expandafter\@ifdefinable\csname #1\endcsname
486     {\global\@namedef{the#1}{\@nameuse{the#2}}}%
487   \global\@namedef{#1}{\@exmpl{#2}{#3}}%
488   \global\@namedef{end#1}{\@endexample}}
489 \def\@exmpl#1#2{%
490   \refstepcounter{#1}%
491   \@ifnextchar[{\@yexmpl{#1}{#2}}{\@xexmpl{#1}{#2}}
492 \def\@xexmpl#1#2{%
493   \@beginexample{#2}{\csname the#1\endcsname}\ignorespaces}
494 \def\@yexmpl#1#2[#3]{%
495   \@opargbeginexample{#2}{\csname the#1\endcsname}{#3}\ignorespaces}
496 \def\@exmplcounter#1{\noexpand\arabic{#1}}
497 \def\@exmplcountersep{.}
498 \def\@beginexample#1#2{%
499   \@nobraektrue\list{}{\setlength{\rightmargin}{\leftmargin}}%
500   \item[{\bfseries #1\ #2}]\mbox{}\\sf}
501 \def\@opargbeginexample#1#2#3{%
502   \@nobraektrue\list{}{\setlength{\rightmargin}{\leftmargin}}%
503   \item[{\bfseries #1\ #2\ (#3)}]\mbox{}\\sf}
504 \def\@endexample{\endlist}
505
506 \newexample{tExample}{\ifnum\language=\languageNGerman Beispiel\else Example\fi}
507
508 \newwrite\struktex@out
509 \newenvironment{example}%
510 {\begingroup% Lets keep the changes local
511   \@bsphack
512   \immediate\openout \struktex@out \jobname.tmp
513   \let\do\@makeother\dospecials\catcode'\~M\active
514   \def\verbatim@processline{%
515     \immediate\write\struktex@out{\the\verbatim@line}}%
516   \verbatim@start}%

```

```

517 {\immediate\closeout\struktex@out\@esphack\endgroup%
518 %
519 % And here comes the part of Tobias Oetiker
520 %
521 \par\small\addvspace{3ex plus 1ex}\vskip -\parskip
522 \noindent
523 \makebox[0.45\linewidth][l]{%
524 \begin{minipage}[t]{0.45\linewidth}
525 \vspace*{-2ex}
526 \setlength{\parindent}{0pt}
527 \setlength{\parskip}{1ex plus 0.4ex minus 0.2ex}
528 \begin{trivlist}
529 \item\input{\jobname.tmp}
530 \end{trivlist}
531 \end{minipage}}%
532 \hfill%
533 \makebox[0.5\linewidth][l]{%
534 \begin{minipage}[t]{0.50\linewidth}
535 \vspace*{-1ex}
536 \verbatiminput{\jobname.tmp}
537 \end{minipage}}
538 \par\addvspace{3ex plus 1ex}\vskip -\parskip
539 }
540
541 \newtoks\verbatim@line
542 \def\verbatim@startline{\verbatim@line{}}
543 \def\verbatim@addtoline#1{%
544 \verbatim@line\expandafter{\the\verbatim@line#1}}
545 \def\verbatim@processline{\the\verbatim@line\par}
546 \def\verbatim@finish{\ifcat$\the\verbatim@line$\else
547 \verbatim@processline\fi}
548
549 \def\verbatimwrite#1{%
550 \@bsphack
551 \immediate\openout \struktex@out #1
552 \let\do\@makeother\dospecials
553 \catcode'\^M\active \catcode'\^^I=12
554 \def\verbatim@processline{%
555 \immediate\write\struktex@out
556 {\the\verbatim@line}}%
557 \verbatim@start}
558 \def\endverbatimwrite{%
559 \immediate\closeout\struktex@out
560 \@esphack}
561
562 \@ifundefined{vrb@catcodes}%
563 {\def\vrb@catcodes{%
564 \catcode'\!12\catcode'\[12\catcode'\]12}}{}
565 \begingroup
566 \vrb@catcodes
567 \lccode'\!='\ \lccode'\[='\ \lccode'\]='\
568 \catcode'\~=\active \lccode'\~='\^^M
569 \lccode'\C='\C
570 \lowercase{\endgroup}

```

```

571 \def\verbatim@start#1{%
572   \verbatim@startline
573   \if\noexpand#1\noexpand~%
574   \let\next\verbatim@
575   \else \def\next{\verbatim@#1}\fi
576   \next}%
577 \def\verbatim@#1~{\verbatim@@#1!end\@nil}%
578 \def\verbatim@@#1!end{%
579   \verbatim@addtoline{#1}%
580   \futurelet\next\verbatim@@@}%
581 \def\verbatim@@@#1\@nil{%
582   \ifx\next\@nil
583     \verbatim@processline
584     \verbatim@startline
585     \let\next\verbatim@
586   \else
587     \def\@tempa##1!end\@nil{##1}%
588     \@temptokena{!end}%
589     \def\next{\expandafter\verbatim@test\@tempa#1\@nil~}%
590     \fi \next}%
591 \def\verbatim@test#1{%
592   \let\next\verbatim@test
593   \if\noexpand#1\noexpand~%
594     \expandafter\verbatim@addtoline
595     \expandafter{\the\@temptokena}%
596     \verbatim@processline
597     \verbatim@startline
598     \let\next\verbatim@
599   \else \if\noexpand#1
600     \@temptokena\expandafter{\the\@temptokena#1}%
601   \else \if\noexpand#1\noexpand[%
602     \let\@tempc\@empty
603     \let\next\verbatim@testend
604   \else
605     \expandafter\verbatim@addtoline
606     \expandafter{\the\@temptokena}%
607     \def\next{\verbatim@#1}%
608     \fi\fi\fi
609     \next}%
610 \def\verbatim@testend#1{%
611   \if\noexpand#1\noexpand~%
612     \expandafter\verbatim@addtoline
613     \expandafter{\the\@temptokena[]}%
614     \expandafter\verbatim@addtoline
615     \expandafter{\@tempc}%
616     \verbatim@processline
617     \verbatim@startline
618     \let\next\verbatim@
619   \else\if\noexpand#1\noexpand[%
620     \let\next\verbatim@@@testend
621   \else\if\noexpand#1\noexpand!%
622     \expandafter\verbatim@addtoline
623     \expandafter{\the\@temptokena[]}%
624     \expandafter\verbatim@addtoline

```

```

625         \expandafter{\@tempc}%
626         \def\next{\verbatim@!}%
627         \else \expandafter\def\expandafter\@tempc\expandafter
628             {\@tempc#1}\fi\fi\fi
629         \next}%
630 \def\verbatim@@testend{%
631     \ifx\@tempc\@currenvir
632     \verbatim@finish
633     \edef\next{\noexpand\end{\@currenvir}}%
634         \noexpand\verbatim@rescan{\@currenvir}}%
635     \else
636     \expandafter\verbatim@addtoline
637     \expandafter{\the\@temptokena[]}%
638     \expandafter\verbatim@addtoline
639     \expandafter{\@tempc}}%
640     \let\next\verbatim@
641     \fi
642     \next}%
643 \def\verbatim@rescan#1#2~{\if\noexpand~\noexpand#2~\else
644     \@warning{Characters dropped after ‘\string\end{#1}’}\fi}}
645
646 \newread\verbatim@in@stream
647 \def\verbatim@readfile#1{%
648     \verbatim@startline
649     \openin\verbatim@in@stream #1\relax
650     \ifeof\verbatim@in@stream
651     \typeout{No file #1.}%
652     \else
653     \@addtofilelist{#1}%
654     \ProvidesFile{#1}[(verbatim)]%
655     \expandafter\endlinechar\expandafter\m@ne
656     \expandafter\verbatim@read@file
657     \expandafter\endlinechar\the\endlinechar\relax
658     \closein\verbatim@in@stream
659     \fi
660     \verbatim@finish
661 }
662 \def\verbatim@read@file{%
663     \read\verbatim@in@stream to\next
664     \ifeof\verbatim@in@stream
665     \else
666     \expandafter\verbatim@addtoline\expandafter{\expandafter\check@percent\next}%
667     \verbatim@processline
668     \verbatim@startline
669     \expandafter\verbatim@read@file
670     \fi
671 }
672 \def\verbatiminput{\begingroup\MacroFont
673     \@ifstar{\verbatim@input\relax}%
674     {\verbatim@input{\frenchspacing\@vobeyspaces}}}%
675 \def\verbatim@input#1#2{%
676     \IfFileExists {#2}{\@verbatim #1\relax
677     \verbatim@readfile{\@filef@und}\endtrivlist\endgroup\@doendpe}%
678     {\typeout {No file #2.}\endgroup}}

```

7 Makefile zur automatisierten Erstellung der Dokumentation und der Tests des `struktex.sty`

Der Umgang mit `.dtx`-Paketen ist wesentlich einfacher, wenn ein Werkzeug für die Automatisierung der wesentlichen Schritte vorliegt. Für Unix/Linux basierte Systeme ist das mit `make` und einem geeigneten `Makefile` einfach zu realisieren. Hier wird der `Makefile` in die Dokumentation integriert, die spezielle Syntax mit Tabulatorzeichen wird durch ein Hilfsprogramm erzeugt, das weiter unten angegeben ist. Auf die Benutzung des `Makefile` wird hier nicht weiter eingegangen, da der erfahrene Benutzer des Werkzeugs diese aus der Datei selbst entnehmen kann.

```
679 #-----
680 # Purpose: generation of the documentation of the struktex package
681 # Notice: this file can be used only with dmake and the option "-B";
682 #         this option lets dmake interpret the leading spaces as
683 #         distinguishing characters for commands in the make rules.
684 #
685 # Rules:
686 #     - all-de:    generate all the files and the (basic) german
687 #                 documentation
688 #     - all-en:    generate all the files and the (basic) english
689 #                 documentation
690 #     - test:      format the examples
691 #     - history:   generate the documentation with revision
692 #                 history
693 #     - develop-de: generate the german documentation with revision
694 #                 history and source code
695 #     - develop-en: generate the english documentation with
696 #                 revision history and source code
697 #     - realclean
698 #     - clean
699 #     - clean-example
700 #
701 # Author:  Jobst Hoffmann, Fachhochschule Aachen, Abt. Juelich
702 # Date:    2003/04/18
703 #-----
704
705 # The texmf-directory, where to install new stuff (see texmf.cnf)
706 # If you don't know what to do, search for directory texmf at /usr.
707 # With teTeX and linux often one of following is used:
708 #INSTALLTEXMF=/usr/TeX/texmf
709 #INSTALLTEXMF=/usr/local/TeX/texmf
710 #INSTALLTEXMF=/usr/share/texmf
711 #INSTALLTEXMF=/usr/local/share/texmf
712 # user tree:
713 #INSTALLTEXMF=$(HOME)/texmf
714 # Try to use user's tree known by kpsewhich:
715 INSTALLTEXMF='kpsewhich --expand-var '$$TEXMFHOME' '
716 # Try to use the local tree known by kpsewhich:
717 #INSTALLTEXMF='kpsewhich --expand-var '$$TEXMFLOCAL' '
718 # But you may set INSTALLTEXMF to every directory you want.
719 # Use following, if you only want to test the installation:
720 #INSTALLTEXMF=/tmp/texmf
721
```

```

722 # If texhash must run after installation, you can invoke this:
723 TEXHASH=texhash
724
725 ##### Edit following only, if you want to change defaults!
726
727 # The directory, where to install *.cls and *.sty
728 CLSDIR=$(INSTALLTEXMF)/tex/latex/jhf/$(PACKAGE)
729
730 # The directory, where to install documentation
731 DOCDIR=$(INSTALLTEXMF)/doc/latex/jhf/$(PACKAGE)
732
733 # The directory, where to install the sources
734 SRCDIR=$(INSTALLTEXMF)/source/latex/jhf/$(PACKAGE)
735
736 # The directory, where to install demo-files
737 # If we have some, we have to add following 2 lines to install rule:
738 #     $(MKDIR) $(DEMODIR); \
739 #     $(INSTALL) $(DEMO_FILES) $(DEMODIR); \
740 DEMODIR=$(DOCDIR)/demo
741
742 # We need this, because the documentation needs the classes and packages
743 # It's not really a good solution, but it's a working solution.
744 TEXINPUTS := $(PWD):$(TEXINPUTS)
745
746 #####
747 #   End of customization section
748 #####
749
750 DVIPS = dvips
751 LATEX = latex
752 PDFLATEX = pdflatex
753
754 # postscript viewer
755 GV = gv
756
757 COMMON_OPTIONS = \OnlyDescription\CodelineNumbered
758 HISTORY_OPTIONS = \RecordChanges
759 DEVELOPER_OPTIONS = \EnableCrossrefs\RecordChanges\AlsoImplementation\CodelineIndex
760
761 PACKAGE = struktex
762
763 all-de: $(PACKAGE).de.pdf
764
765 all-en: $(PACKAGE).en.pdf
766
767 # strip off the comments from the package
768 $(PACKAGE).sty $(PACKAGE)-test-*.tex: $(PACKAGE).dtx
769
770 $(PACKAGE).sty $(PACKAGE)-test-*.tex: $(PACKAGE).ins
771 +$(LATEX) $<
772
773 # generate the documentation
774 $(PACKAGE).dvi: $(PACKAGE).sty
775

```

```

776 $(PACKAGE).de.dvi: $(PACKAGE).dtx
777 +$(LATEX) "\AtBeginDocument{$(COMMON_OPTIONS)}\input{<}\"
778 +$(LATEX) "\AtBeginDocument{$(COMMON_OPTIONS)}\input{<}\"
779 +mv $(<:.dtx=.dvi) $(<:.dtx=.de.dvi)
780
781 $(PACKAGE).de.pdf: $(PACKAGE).dtx
782 +$(PDFLATEX) "\AtBeginDocument{$(COMMON_OPTIONS)}\input{<}\"
783 +$(PDFLATEX) "\AtBeginDocument{$(COMMON_OPTIONS)}\input{<}\"
784 +mv $(<:.dtx=.pdf) $(<:.dtx=.de.pdf)
785
786 $(PACKAGE).en.dvi: $(PACKAGE).dtx
787 +$(LATEX) "\AtBeginDocument{$(COMMON_OPTIONS)}\def\selectlanguageEnglish{}\input{<}\"
788 +$(LATEX) "\AtBeginDocument{$(COMMON_OPTIONS)}\def\selectlanguageEnglish{}\input{<}\"
789 +mv $(<:.dtx=.dvi) $(<:.dtx=.en.dvi)
790
791 $(PACKAGE).en.pdf: $(PACKAGE).dtx
792 +$(PDFLATEX) "\AtBeginDocument{$(COMMON_OPTIONS)}\def\selectlanguageEnglish{}\input{<}\"
793 +$(PDFLATEX) "\AtBeginDocument{$(COMMON_OPTIONS)}\def\selectlanguageEnglish{}\input{<}\"
794 +mv $(<:.dtx=.pdf) $(<:.dtx=.en.pdf)
795
796 # generate the documentation with revision history (only german)
797 history: $(PACKAGE).dtx
798 +$(LATEX) "\AtBeginDocument{$(COMMON_OPTIONS)}$(HISTORY_OPTIONS)}\input{<}\"
799 +$(LATEX) "\AtBeginDocument{$(COMMON_OPTIONS)}$(HISTORY_OPTIONS)}\input{<}\"
800 +makeindex -s gind.ist $(PACKAGE).idx
801 +makeindex -s gglo.ist -o $(PACKAGE).gls -t $(PACKAGE).glg $(PACKAGE).glo
802 +$(LATEX) "\AtBeginDocument{$(COMMON_OPTIONS)}$(HISTORY_OPTIONS)}\input{<}\"
803
804 # generate the documentation for the developer (revision history always
805 # in german)
806 develop-de: $(PACKAGE).dtx
807 +$(LATEX) "\AtBeginDocument{$(HISTORY_OPTIONS)}$(DEVELOPER_OPTIONS)}\input{<}\"
808 +$(LATEX) "\AtBeginDocument{$(HISTORY_OPTIONS)}$(DEVELOPER_OPTIONS)}\input{<}\"
809 +makeindex -s gind.ist $(PACKAGE).idx
810 +makeindex -s gglo.ist -o $(PACKAGE).gls -t $(PACKAGE).glg $(PACKAGE).glo
811 +$(LATEX) "\AtBeginDocument{$(HISTORY_OPTIONS)}$(DEVELOPER_OPTIONS)}\input{<}\"
812 ifneq (,$(findstring pdf,$(LATEX)))
813 +mv $(<:.dtx=.pdf) $(<:.dtx=.de.pdf)
814 else
815 +mv $(<:.dtx=.dvi) $(<:.dtx=.de.dvi)
816 endif
817
818 develop-en: $(PACKAGE).dtx
819 +$(LATEX) "\AtBeginDocument{$(COMMON_OPTIONS)}$(DEVELOPER_OPTIONS)}\def\selectlanguageEnglish{
820 +$(LATEX) "\AtBeginDocument{$(COMMON_OPTIONS)}$(DEVELOPER_OPTIONS)}\def\selectlanguageEnglish{
821 +makeindex -s gind.ist $(PACKAGE).idx
822 +makeindex -s gglo.ist -o $(PACKAGE).gls -t $(PACKAGE).glg $(PACKAGE).glo
823 +$(LATEX) "\AtBeginDocument{$(COMMON_OPTIONS)}$(DEVELOPER_OPTIONS)}\def\selectlanguageEnglish{
824 ifneq (,$(findstring pdf,$(LATEX)))
825 +mv $(<:.dtx=.pdf) $(<:.dtx=.en.pdf)
826 else
827 +mv $(<:.dtx=.dvi) $(<:.dtx=.en.dvi)
828 endif
829

```

```

830 # format the example/test files
831 test:
832 for i in `seq 1 3`; do \
833     f=$(PACKAGE)-test-$$i; \
834     echo file: $$f; \
835     $(LATEX) $$f; \
836     $(DVIPS) -o $$f.ps $$f.dvi; \
837     $(GV) $$f.ps \&; \
838 done
839
840 install: $(PACKAGE).dtx $(PACKAGE).dvi
841 [ -d $(CLSDIR) ] || mkdir -p $(CLSDIR)
842 [ -d $(DOCDIR) ] || mkdir -p $(DOCDIR)
843 [ -d $(SRCDIR) ] || mkdir -p $(SRCDIR)
844 cp $(PACKAGE).sty $(CLSDIR)
845 cp $(PACKAGE).dvi $(DOCDIR)
846 cp $(PACKAGE).ins $(SRCDIR)
847 cp $(PACKAGE).dtx $(SRCDIR)
848 cp $(PACKAGE)-test-*.tex $(SRCDIR)
849 cp LIESMICH $(SRCDIR)
850 cp README $(SRCDIR)
851 cp THIS-IS-VERSION-$(VERSION) $(SRCDIR)
852
853 uninstall:
854 rm -f $(CLSDIR)/$(PACKAGE).sty
855 rm -fr $(DOCDIR)
856 rm -fr $(SRCDIR)
857
858 pack: $(PACKAGE).de.pdf $(PACKAGE).en.pdf $(PACKAGE).dtx $(PACKAGE).ins \
859 LIESMICH README
860 + tar cvfz $(PACKAGE).tgz $^
861
862 clean:
863 -rm -f *.log *.aux *.brf *.idx *.ilg *.ind
864 -rm -f *.glg *.glo *.gls *.lof *.lot *.out *.toc *.tmp *~
865 -rm *.mk *.makemake
866
867 realclean: clean
868 -rm -f *.sty *.cls *.ps *.dvi *.pdf
869 -rm -f *test* getversion.* Makefile
870
871 clean-test:
872 rm $(PACKAGE)-test-*. * # this $-sign is needed for font-locking in XEmacs only

```

Die folgende Zeile, die nach latex `struktex.ins` als Datei `struktex.makemake` vorliegt, kann mit dem Kommando

```
sh struktex.makemake
```

dazu benutzt werden, die obige Datei in ein Format umzusetzen, das von üblichen make-Programmen wie dem GNU make verarbeitet werden kann.

```
873 sed -e "s/^ /@/g" | tr '@' '\011' " struktex.mk > Makefile
```

Die folgende Datei dient allein dazu, die Version des Paketes zu ermitteln.

```

874 \documentclass[english]{ltxdoc}
875 \nofiles
876 \usepackage{struktex}
877 \GetFileInfo{struktex.sty}
878 \typeout{VERSION \fileversion}
879 \begin{document}
880 \end{document}

```

8 Stil Datei zur einfachen Eingabe von Struktogrammen beim Arbeiten mit dem (X)emacs und AUCT_EX

Der (X)emacs und das Paket AUCT_EX (<http://www.gnu.org/software/auctex/>) bilden ein mächtiges Werkzeug beim Erstellen von T_EX/L^AT_EX-Dateien. Wenn es eine passende Stildatei für ein L^AT_EX-Paket wie das hier vorliegende St_ukT_EX gibt, wird die Eingabe von Quelltext durch automatisiertes Vervollständigen und viele andere Hilfsmittel sehr erleichtert. Im folgenden wird eine derartige Stildatei bereitgestellt; sie muss nach ihrer Erzeugung noch an eine entsprechende Stelle kopiert werden.

Diese Datei ist zum jetzigen Zeitpunkt noch in der Entwicklungsphase, d. h. man kann mit ihr arbeiten, aber es fehlen noch ein paar Dinge wie das *font locking*, die Anzahl der automatisch eingefügten `\switch` Kommandos sollte nicht fest gleich eins sein, sondern von der Anzahl der eingegebenen Fille abhngig sein.

```

881 ;; struktex.el --- AUCTeX style for 'struktex.sty'
882
883 ;; Copyright (C) 2006 Free Software Foundation, Inc.
884
885 ;; Author: J. Hoffmann <j.hoffmann@fh-aachen.de>
886 ;; Maintainer: j.hoffmann@fh-aachen.de
887 ;; Created: 2006/01/17
888 ;; Keywords: tex
889
890 ;;; Commentary:
891 ;; This file adds support for 'struktex.sty'
892
893 ;;; Code:
894 (TeX-add-style-hook
895  "struktex"
896  (lambda ()
897    ;; Add declaration to the list of environments which have an optional
898    ;; argument for each item.
899    (LaTeX-add-environments
900     "centernss"
901     '("struktogramm" LaTeX-env-struktogramm)
902     '("declaration" LaTeX-env-declaration))
903    (TeX-add-symbols
904     '("PositionNSS" 1)
905     '("assert" [ "Height" ] "Assertion")
906     '("assign" [ "Height" ] "Statement")
907     "StrukTeX"

```

```

908 '("case" TeX-mac-case)
909 "switch" "condition"
910 "caseend"
911 '("declarationtitle" "Title")
912 '("description" "Name" "Meaning")
913 "emptyset"
914 '("exit" [ "Height" ] "What" )
915 '("forever" TeX-mac-forever)
916 "foreverend"
917 '("ifthenelse" TeX-mac-ifthenelse)
918 "ifend"
919 "change"
920 "sProofOn"
921 "sProofOff"
922 '("until" TeX-mac-until)
923 "untilend"
924 '("while" TeX-mac-while)
925 "whileend"
926 '("return" [ "Height" ] "Return value")
927 '("sub" [ "Height" ] "Task")
928 '("CenterNssFile" TeX-arg-file)
929 '("centernssfile" TeX-arg-file))
930 (TeX-run-style-hooks
931 "pict2e"
932 "emlines2"
933 "curves"
934 "struktxp"
935 "struktxf"
936 "ifthen")
937 ;; Filling
938 ;; Fontification
939 ))
940
941 (defun LaTeX-env-struktogramm (environment)
942 "Insert ENVIRONMENT with width, height specifications and optional title."
943 (let ((width (read-string "Width: "))
944       (height (read-string "Height: "))
945       (title (read-string "Title (optional): ")))
946   (LaTeX-insert-environment environment
947     (concat
948       (format "(%s,%s)" width height)
949       (if (not (zerop (length title)))
950           (format "[%s]" title))))))
951
952 (defun LaTeX-env-declaration (environment)
953 "Insert ENVIRONMENT with an optional title."
954 (let ((title (read-string "Title (optional): ")))
955   (LaTeX-insert-environment environment
956     (if (not (zerop (length title)))
957         (format "[%s]" title))))))
958
959 (defun TeX-mac-case (macro)
960 "Insert \case with all arguments.
961 These are optional height and the required arguments slope, number of cases,
```

```

962 condition, and the text of the first case"
963 (let ((height (read-string "Height (optional): "))
964       (slope (read-string "Slope: "))
965       (number (read-string "Number of cases: "))
966       (condition (read-string "Condition: "))
967       (text (read-string "Text: ")))
968 (insert (concat (if (not (zerop (length height)))
969                  (format "[%s]" height))
970              (format "{%s}{%s}{%s}{%s}"
971                    slope number condition text)))
972 (end-of-line)
973 (newline-and-indent)
974 (newline-and-indent)
975 (insert "\\switch{ }")
976 (end-of-line)
977 (newline-and-indent)
978 (newline-and-indent)
979 (insert "\\caseend")))
980
981 (defun TeX-mac-forever (macro)
982 "Insert \forever-block with all arguments.
983 This is only the optional height"
984 (let ((height (read-string "Height (optional): "))
985       (insert (if (not (zerop (length height)))
986                 (format "[%s]" height)))
987       (end-of-line)
988       (newline-and-indent)
989       (newline-and-indent)
990       (insert "\\foreverend")))
991
992 (defun TeX-mac-ifthenelse (macro)
993 "Insert \ifthenelse with all arguments.
994 These are optional height and the required arguments left slope, right slope,
995 condition, and the possible values of the condition"
996 (let ((height (read-string "Height (optional): "))
997       (lslope (read-string "Left slope: "))
998       (rslope (read-string "Right slope: "))
999       (condition (read-string "Condition: "))
1000      (conditionvl (read-string "Condition value left: "))
1001      (conditionvr (read-string "Condition value right: ")))
1002 (insert (concat (if (not (zerop (length height)))
1003                  (format "[%s]" height))
1004              (format "{%s}{%s}{%s}{%s}{%s}"
1005                    lslope rslope condition conditionvl conditionvr)))
1006 (end-of-line)
1007 (newline-and-indent)
1008 (newline-and-indent)
1009 (insert "\\change")
1010 (end-of-line)
1011 (newline-and-indent)
1012 (newline-and-indent)
1013 (insert "\\ifend")))
1014
1015 (defun TeX-mac-until (macro)

```

```

1016 "Insert \until with all arguments.
1017 These are the optional height and the required argument condition"
1018 (let ((height (read-string "Height (optional): "))
1019       (condition (read-string "Condition: ")))
1020   (insert (concat (if (not (zerop (length height)))
1021                     (format "[%s]" height))
1022                 (format "{%s}" condition)))
1023   (end-of-line)
1024   (newline-and-indent)
1025   (newline-and-indent)
1026   (insert "\\untilend"))
1027
1028 (defun TeX-mac-while (macro)
1029 "Insert \while with all arguments.
1030 These are the optional height and the required argument condition"
1031 (let ((height (read-string "Height (optional): "))
1032       (condition (read-string "Condition: ")))
1033   (insert (concat (if (not (zerop (length height)))
1034                     (format "[%s]" height))
1035                 (format "{-%s-}" condition)))
1036   (end-of-line)
1037   (newline-and-indent)
1038   (newline-and-indent)
1039   (insert "\\whileend"))
1040
1041 (defvar LaTeX-struktex-package-options '("curves" "draft" "emlines" "final"
1042                                         "pict2e" "anygradient" "verification"
1043                                         "nofiller")
1044 "Package options for the struktex package."
1045
1046 ;;; struktex.el ends here.

```

9 Die Implementation

Der `struktex.sty` stellt eine einfache Implementation der beschriebenen Makros dar; er sollte prinzipiell mit anderen Makropaketen verträglich sein. Die Gültigkeit von Makros wurde im Regelfall auf einen möglichst kleinen Bereich beschränkt. Um Seiteneffekte mit anderen *packages* so weit wie möglich zu vermeiden, werden alle intern genutzten Makros mit dem Suffix `@nss` versehen.

9.1 Allgemeines

Als erstes wird getestet, ob das Paket `struktex.sty` noch nicht geladen wurde. Wenn das der Fall ist, wird weitergearbeitet, ansonsten wird das Lesen sofort abgebrochen.

```
1047 (*struktex)
1048 \@ifundefined{StrukTeX}{}{\endinput}
```

Dies ist ein Paket, das für L^AT_EX₂e entwickelt wurde, es wird daher eine entsprechende Fehlermeldung ausgegeben, wenn ein anderes Format benutzt wird, was nicht zu Fehlern führen sollte; zusätzlich wird das Paket `ifthen.sty` geladen, falls es noch nicht anderweitig angefordert wurde. Nun wird das Einlesen des Makropakets mit der aktuellen Versionsnummer gemeldet.

```
1049 \RequirePackage{ifthen}
```

StrukTeX benötigt zwei weitere Pakete, die zum Lieferumfang gehören. Diese werden hier geladen.

```
1050 \RequirePackage{struktxf}
1051 \RequirePackage{strukt xp}
```

Ab der Version 5.3a kennt StrukTeX eine Option „`emlines`“, die es ermöglicht, Geraden beliebiger Steigung zu zeichnen; später kann auch das Paket `curves` zum Zeichnen beliebiger Steigungen eingesetzt werden, ab der Version 8.0a wird auch das Paket `pict2e` unterstützt, das dazugehörige Paket wird hier geladen.

```
1052 \newboolean{curves}
1053 \newboolean{emlines}
1054 \newboolean{pictIIe}
1055 \newboolean{anygradient}
1056 \DeclareOption{curves}%
1057 {%
1058     \setboolean{anygradient}{true}
1059     \setboolean{curves}{true}
1060     \setboolean{emlines}{false}
1061     \setboolean{pictIIe}{false}
1062 }
1063 \DeclareOption{emlines}%
1064 {%
1065     \setboolean{anygradient}{true}
1066     \setboolean{curves}{false}
1067     \setboolean{emlines}{true}
1068     \setboolean{pictIIe}{false}
1069 }
1070 \DeclareOption{pict2e}%
1071 {%
1072     \setboolean{anygradient}{true}
1073     \setboolean{curves}{false}
```

```

1074 \setboolean{emlines}{false}
1075 \setboolean{pictIIE}{true}
1076 }

```

Ab der Version 7.0a kennt \StrukTeX eine Option „`verification`“, die ein zusätzliches Element definiert: die `\assert`-Box, die in [Fut89] eingeführt wird.

```

1077 \newboolean{verification}
1078 \DeclareOption{verification}%
1079 {%
1080 \setboolean{verification}{true}
1081 }

```

Ab der Version 8.2a kennt \StrukTeX eine Option „`nofiller`“, die die Darstellung von Leerraum beeinflusst: ohne diese Option wird in Alternativen ein \emptyset -Zeichen ausgegeben, wenn der Platz ausreichend ist. Dies führt zu einer uneinheitlichen Darstellung im Vergleich mit dem Leerraum, der von einem `\case`-Block erzeugt wird, da hier der Leerraum nicht in entsprechende Weise angezeigt wird – dies wäre nur mit einer aufwendigen Listenverwaltung möglich. Angabe der Option führt dazu, dass das \emptyset -Zeichen einheitlich weggelassen wird.

```

1082 \newboolean{filler}
1083 \setboolean{filler}{true}
1084 \DeclareOption{nofiller}%
1085 {%
1086 \setboolean{filler}{false}
1087 }

```

Ab der Version 8.3 kennt \StrukTeX die Optionen „`draft`“ und „`final`“, die zwischen dem Entwurf und der endgültigen Version des Struktogramms umschalten. Entwurfsmodus ist `\sProofOn` gesetzt, für die endgültige Version wird `\sProofOff` gesetzt.

```

1088 \let\ifdraft@nss\iffalse
1089 \let\iffinal@nss\iftrue
1090 \DeclareOption{draft}%
1091 {%
1092 \let\ifdraft@nss\iftrue
1093 \let\iffinal@nss\iffalse
1094 }
1095 \DeclareOption{final}%
1096 {%
1097 \let\ifdraft@nss\iffalse
1098 \let\iffinal@nss\iftrue
1099 }

```

Nun werden andere Pakete geladen und die Optionen verarbeitet.

```

1100 \ProcessOptions
1101 \ifcurves%
1102 \RequirePackage{curves}
1103 \else\ifemlines
1104 \RequirePackage{emlines2}
1105 \else\ifpictIIE
1106 \RequirePackage{pict2e}
1107 \fi\fi\fi

```

\StrukTeX Das Logo des Paketes, anhand dessen späteres erneutes Einlesen unterbunden wird:

```

1108 \def\StrukTeX{S\kern-.05emT\kern-.05em\raise.5ex\hbox{r}%
1109         \kern-.1667emu\kern-.05em\lower.5ex\hbox{k}%
1110         \kern-.2emT\kern-.1667em\lower.7ex\hbox{E}\kern-.125emX}

```

9.2 Diverse Makros zur Vorbereitung

Zunächst werden die zusätzlich benötigten Fonts bereitgestellt. Da diese Fonts auch in anderem Zusammenhang benötigt werden, wird hierfür ein eigenes Paket definiert.

```

1111 \</struktex>
1112 \*struktexf
1113 \@ifundefined{nat}{\endinput}
1114 \DeclareSymbolFont{italics}{\encodingdefault}{\rmdefault}{m}{it}%
1115 \DeclareSymbolFont{AMSb}{U}{msb}{m}{n}
1116 \DeclareSymbolFontAlphabet{\mathbb}{AMSb}

```

`\nat` Die folgenden Makros dienen dem vereinfachten Schreiben von Mengen:

```

\integer 1117 \def\nat{\mathbb N}
\real 1118 \def\integer{\mathbb Z}
\complex 1119 \def\real{\mathbb R}
1120 \def\complex{\mathbb C}

```

`\btt` Um fetten Schreibmaschinensatz erzeugen zu können, gibt es den Font `\btt`, der als Erweiterung der `cmTT`-Font-Familie definiert ist. Um auf diesen Font auch ohne zusätzliches Einbinden dieses *packages* zugreifen zu können, kann man die entsprechenden Deklarationen in den Dateien `ot1cmTT.fd` bzw. `t1cmTT.fd` durch die folgende ersetzen.⁵

```

1121 \DeclareFontFamily{OT1}{cmbtt}{}
1122 \DeclareFontShape{OT1}{cmTT}{bx}{n}{
1123   <-8> cmbtt8 <9> cmbtt9 <10-> cmbtt10
1124 }{}
1125
1126 \def\btt%
1127 {%
1128   \fontencoding{\encodingdefault}\fontfamily{cmbtt}\fontseries{bx}%
1129   \fontshape{n}\selectfont%
1130 }

```

`\MathItalics` Die folgenden Makros dienen dem Umschalten zwischen verschiedenen Formen der *italics* im Mathematik-Modus. Sie wurden aus [Rah92] übernommen. Als Standard wird `\MathItalics` genommen, um eine schönere Darstellung von langen Variablenamen zu erzielen.

```

\MathNormal
1131 \def\@setmcodes#1#2#3{\count0=#1 \count1=#3
1132   \loop \global\mathcode\count0=\count1 \ifnum \count0<#2
1133     \advance\count0 by1 \advance\count1 by1 \repeat}
1134 \def\MathItalics%
1135 {%
1136   \@setmcodes{‘A}{‘Z}{"7\hexnumber@\symitalics41}
1137   \@setmcodes{‘a}{‘z}{"7\hexnumber@\symitalics61}
1138 }
1139 \def\MathNormal%

```

⁵Eine Alternative ist die Benutzung des Luxi Mono Fonts.

```

1140 {%
1141     \@setmcodes{‘A}{‘Z}{“7141}
1142     \@setmcodes{‘a}{‘z}{“7161}
1143 }
1144 \MathItalics
1145 \</struktxf>
1146 \*struktex

```

9.3 Makros zur Darstellung von Variablen, Schlüsselwörtern und anderen programmierspezifischen Details

Struktogramme enthalten manchmal direkt zu programmierenden Code. Um hier ein einheitliches Aussehen zu erreichen, sind die folgenden Makros definiert worden. Um diese Makros auch in anderem Zusammenhang nutzen zu können, sind sie zu einem eigenen *package* `strukt xp.sty` zusammengefasst worden.

```

1147 \</struktex>
1148 \*strukt xp
1149 \@ifundefined{pVariable}{}{\<endinput>}

```

`\pExpFont@nss` Die drei Makros `\pExpFont@nss`, `\pKeyFont@nss` und `\pCommentFont@nss` dienen der Vorbesetzung der Fonts zur Formatierung von Variablen, Schlüsselwörtern und Kommentar. Diese kann mit dem Kommando `\pFonts` undefiniert werden.

```

1150 \newcommand{\pExpFont@nss}{\small\sffamily}
1151 \newcommand{\pKeyFont@nss}{\small\sffamily\bfseries}
1152 \newcommand{\pCommentFont@nss}{\small\sffamily\slshape}

```

`\pFonts` Der Makro `\pFonts` dient dem Umsetzen der Vorbesetzungen der Fonts zum Setzen von Variablennamen, Schlüsselwörtern und Kommentar:

```

1153 \newcommand{\pFonts}[3]%
1154 {%
1155     \def\argi@nss{#1} \def\argii@nss{#2} \def\argiii@nss{#3}
1156     \ifx\argi@nss\empty\else%
1157         \renewcommand{\pExpFont@nss}{#1}%
1158     \fi%
1159     \ifx\argii@nss\empty\else%
1160         \renewcommand{\pKeyFont@nss}{#2}%
1161     \fi%
1162     \ifx\argiii@nss\empty\else%
1163         \renewcommand{\pCommentFont@nss}{#3}%
1164     \fi%
1165 }

```

Zum Setzen von Programmtexten innerhalb von Struktogrammen wird unterschieden zwischen Ausdrücken und (Schlüssel-)Wörtern. Ein Ausdruck (zu verstehen im Sinne eines Ausdrucks einer Programmiersprache) wird in einem festen Font gesetzt, der durch `\pExpFont@nss` aktuell bestimmt wird. Dieses gilt in gleicher Weise für (Schlüssel-)Wörter, dabei werden aber die Schlüsselwörter, die in einer entsprechenden Liste definiert sind, fett gedruckt. Sowohl Ausdrücke als auch (Schlüssel-)wörter dürfen Sonderzeichen enthalten:

1. der Unterstrich „_“,
2. das Dach „^“,

3. das kaufmännische Und „&“ und

4. das Doppelkreuz „#“.

`\pExpression` Leerzeichen werden beachtet, so dass ganze Anweisungen geschrieben werden können. Das Argument kann erst nach der Umdefinition der `catcodes` von einem internen Makro ausgewertet werden.

Zur Erleichterung des Schreibens wird mit `\let` ein alternativer Zugriff auf `\pVariable` geschaffen.

```
1166 \newcommand{\pExpression}%  
1167 {%  
1168   \bgroup%
```

Zunächst werden die benötigten `catcodes` auf Sonstige – *other* – umgesetzt, zur Erinnerung:

<i>catcode</i>	Bedeutung
1	Gruppenanfang
2	Gruppenende
4	Tabulatorzeichen
8	Index
12	Sonstige

In einer Variablen sind die folgenden Sonderzeichen erlaubt:

1. der Unterstrich „_“,
2. das Dach „^“ und
3. das kaufmännische Und „&“.

```
1169   \catcode'\_ =12 \catcode'\^ =12 \catcode'\& =12 %  
1170   \pUseExpFont@nss%  
1171 }  
1172 \let\pExp=\pExpression
```

`\pVariable` Die Implementation von `\pKeyword` entspricht der von `\pVariable`. Die hier neu benutzten Fonts (cmbtt..) werden durch den `struktxf.sty` bereitgestellt. Ansonsten wird wie bei `\pVariable` verfahren: Zusätzlich werden Makros `\pVar` sowie `\pKeyword` und `\pKey` mit der gleichen Bedeutung definiert. Diese dienen allein der Rückwärtskompatibilität.

```
1173 \newcommand{\pVariable}%  
1174 {%  
1175   \bgroup%  
1176   \catcode'\_ =13 \let\_ \textunderscore  
1177   \catcode'\# =12 \catcode'\^ =12 \catcode'\& =12 %  
1178   \pVariabl@%  
1179 }  
1180 \let\pVar=\pVariable  
1181 \let\pKeyword=\pVariable  
1182 \let\pKey=\pVariable
```

`\pVariabl@` Die Liste der Schlüsselwörter wird nach den verschiedenen Anwendungsgebieten unterschieden.⁶

⁶Die Python-Erweiterung erfolgte nach einem Hinweis von Ludger Humbert, Universität Dortmund, von dem auch die Liste der Schlüsselwörter stammt, vielen Dank!

```

1183 \newcommand{\pVariabl@}[1]
1184 {%
1185     \def\arg{#1}%
1186     \ifx\pLanguage@nss\Cee@nss%
1187         \CheckForKeyword@nss{auto,break,case,char,const,continue,default,do,double,%
1188         else,enum,extern,float,for,goto,if,int,long,register,return,%
1189         short,signed,sizeof,static,struct,switch,typedef,union,unsigned,%
1190         void,volatile,while}%
1191     \else\ifx\pLanguage@nss\Java@nss%
1192         \CheckForKeyword@nss{abstract,boolean,break,byte,case,catch,char,class,const,%
1193         continue,default,do,double,else,extends,final,finally,float,for,%
1194         goto,if,implements,import,instanceof,int,interface,long,native,%
1195         new,null,package,private,protected,public,return,short,static,%
1196         super,switch,synchronized,this,throw,throws,transient,try,void,%
1197         volatile,while,true,false}%
1198     \else\ifx\pLanguage@nss\Pascal@nss%
1199         \CheckForKeyword@nss{alfa,and,array,begin,boolean,byte,case,char,const,div,do,%
1200         downto,else,end,false,file,for,function,get,goto,if,in,integer,%
1201         label,maxint,mod,new,not,of,or,pack,packed,page,program,put,%
1202         procedure,read,readln,real,record,repeat,reset,rewrite,set,text,%
1203         then,to,true,type,unpack,until,var,while,with,write,writeln}%
1204     \else\ifx\pLanguage@nss\Python@nss%
1205         \CheckForKeyword@nss{and,assert,break,class,continue,def,del,elif,else,%
1206         except,exec,finally,for,from,global,if,import,in,is,lambda,not,or,%
1207         pass,print,raise,return,try,while,yield}%
1208     \else\ifx\pLanguage@nss\LaTeX@nss%
1209         \CheckForKeyword@nss{center,description,enumerate,figure,itemize,
1210         list,quote,tabbing,tabular,table,}%
1211     \else%
1212         \CheckForKeyword@nss{}%
1213     \fi\fi\fi\fi\fi%
1214     \egroup%
1215 }

```

`\pLanguage` Die Liste der Schlüsselwörter wird nach den verschiedenen Anwendungsgebieten unterschieden.

```

\Java@nss 1216 \newcommand{\pLanguage}[1]{\gdef\pLanguage@nss{#1}}
\Pascal@nss 1217 \def\Cee@nss{C}
\LaTeX@nss 1218 \def\Java@nss{Java}
\Python@nss 1219 \def\Pascal@nss{Pascal}
1220 \def\LaTeX@nss{LaTeX}
1221 \def\Python@nss{Python}
1222 \def\OSII@nss{OS/2}

```

`\CheckForKeyword@nss` Die Liste der Schlüsselwörter wird der Reihe nach durchlaufen und mit dem aktuellen Argument `\arg` verglichen.

```

1223 \def\CheckForKeyword@nss#1%
1224 {%
1225     \let\exec\pExpFont@nss%
1226     \def\endList{!}%
1227     \def\yyI##1,%
1228     {%
1229         \def\token{##1}%
1230         \ifx\token\endList%

```

```

1231         \ifmode\else\let\mbox=\relax\fi%
1232         \mbox{\exec\selectfont\arg}%
1233     \else%
1234         \ifx\token\arg%
1235             \let\exec\pKeyFont@nss%
1236         \fi%
1237         \expandafter\yyI%
1238     \fi%
1239 }%
1240 \yyI#1,!,%
1241 }

```

`\pComment` Zur Auszeichnung von Kommentar.

```

1242 \def\pComment%
1243 {%
1244     \bgroup%
1245     \pCommentFont@nss%
1246     \let\next=%
1247 }

```

`\pUseKeyFont@nss` Damit Unterstriche und ähnliche Sonderzeichen korrekt dargestellt werden können,
`\pUseExpFont@nss` ist vor der Benutzung eines bestimmten Fonts zu überprüfen, ob die gewünschten Zeichen in dem betreffenden Font überhaupt zur Verfügung stehen.

Es wird stets auf den Textmodus umgeschaltet, was unter Einsatz des `amstex`-Paketes noch besser durch das Kommando `\text` geschehen könnte. Es können Probleme mit dem Zeichen \textasciitilde T ⁷ auftreten, in diesem Fall ist das Zeichen durch \textasciitilde Y ⁸ zu ersetzen

```

1248 {
1249     \catcode'\#=6%
1250     \gdef\pUseKeyFont@nss#1%
1251     {%
1252         \ifmode\else\let\mbox=\relax\fi%
1253         \mbox{\pKeyFont@nss{#1}}\egroup%
1254     }
1255     \gdef\pUseExpFont@nss#1%
1256     {%
1257         \ifmode\else\let\mbox=\relax\fi%
1258         \mbox{\pExpFont@nss{#1}}\egroup%
1259     }
1260 }

```

`\pTrue@nss` Die Vorbesetzungen der logischen Werte `\pTrue` und `\pFalse`:

```

\pFalse@nss 1261 \newcommand{\pTrue@nss}{\texttt{WAHR}}
              1262 \newcommand{\pFalse@nss}{\texttt{FALSCH}}

```

`\pTrue` Um einheitliches Aussehen logischer Werte zu erzielen, werden entsprechende Makros mit den obigen Vorbesetzungen definiert.

```

1263 \newcommand{\pTrue}{\pTrue@nss}
1264 \newcommand{\pFalse}{\pFalse@nss}

```

⁷Dies funktioniert auf einem Linux-System

⁸getestet unter OS/2

`\pBoolValue` Mit `\pBoolValue` können die voreingestellten Werte von `\pTrue` und `\pFalse` verändert werden; `\sBoolValue` ist der alternative Aufruf zwecks Kompatibilität mit alten Struktogrammen.

```
1265 \newcommand{\pBoolValue}[2]%
1266 {%
1267     \renewcommand{\pTrue@nss}{#1}%
1268     \renewcommand{\pFalse@nss}{#2}%
1269 }
1270 \let\sBoolValue=\pBoolValue
```

`\sVar` Zur Darstellung von Variablen. Hier nur noch zwecks Kompatibilität zu früheren Versionen des `struktex.sty` aufgeführt.

```
\sTrue 1271 \let\sVar=\pVariable
\sFalse 1272 \let\sKey=\pKeyword
1273 \let\sTrue=\pTrue
1274 \let\sFalse=\pFalse
1275 </struktex>
1276 <*struktex>
```

9.4 Belegung von Registern

`\savelength@nss` `\savelength@nss` dient dazu, die vor dem Struktogramm gültige Längeneinheit zwischenzuspeichern, um nach dem Struktogramm den alten Wert wiederherstellen zu können.

```
1277 \newdimen\savelength@nss
```

`\gx@nss` `\gx@nss`, `\gy@nss` und `\gsize@nss` enthalten die *globalen* Werte für die x- und y-Koordinate und die Breite des Struktogramms; zur Vereinfachung der Anwendung `\gsize@nss` wird in allen Fällen mit Zählregistern gearbeitet.

```
1278 \newcount\gx@nss \newcount\gy@nss \newcount\gsize@nss
```

`\param@nss` `\param@nss` enthält den Wert, der als optionales Argument für die Breite oder Höhe eines Struktogramms angegeben wird.

```
1279 \countdef\param@nss=199
```

`\x@nss` Es werden einige weitere Variablen benötigt, die alle zentral an dieser Stelle definiert werden: `\x@nss`, `\y@nss` und `\xsize@nss` als lokale Werte für die *x*- und *y*-Koordinate und die Breite eines (Unter-)Struktogramms; `\ydepth@nss` gibt die „Tiefe“ eines Struktogrammelementes an. Die Variablen `\xx@nss`, `\yy@nss`, `\tempx@nss`, `\tempxx@nss`, `\tempxxx@nss`, `\tempy@nss`, `\tempyy@nss` und `\tempyyy@nss` haben unterschiedliche temporäre Bedeutungen, im Regelfall deutet ein *x* im Namen auf horizontalen und ein *y* auf vertikalen Zusammenhang hin.

```
\tempxxx@nss 1280 \countdef\x@nss=220 \countdef\y@nss=221 \countdef\xsize@nss=222
\tempy@nss 1281 \countdef\ydepth@nss=223
\tempyy@nss 1282 \countdef\xx@nss=224 \countdef\yy@nss=225
\tempyyy@nss 1283 \countdef\tempx@nss=226 \countdef\tempxx@nss=227
\ydepth@nss 1284 \countdef\tempy@nss=229 \countdef\tempyy@nss=230
1285 \ifpictIle%
1286 \countdef\tempxxx@nss=228 \countdef\tempyyy@nss=231
1287 \fi
```

`\TextBox@nss` `\TextBox@nss` ist die Box, die den Text in einem Struktogrammblock enthält, ihre tatsächliche Höhe, Überlänge und Unterlänge wird in `\TextBoxHt@nss` abgelegt.

```
1288 \newbox\TextBox@nss \newdimen\TextBoxHt@nss
```

9.5 Die einzelnen Makros

`\filler@nss` Beim Setzen eines Struktogrammes entsteht teilweise Leerraum, der mit `\filler@nss` aufgefüllt wird: „ \emptyset “. Eine Alternative wäre „ \emptyset “, was mit „`\mathchar"023B`“ erzeugt wird. Der Leerraum wird allerdings nicht gefüllt, wenn die Option `nofiller` gesetzt wird. Will man die globale Einstellung lokal umgehen, so kann man einfach die boolesche Variable `filler` auf `true` setzen: `\fillertrue`.

```
1289 \DeclareMathSymbol\varnothing{\mathord}{AMSB}{"3F}
1290 \let\emptyset\varnothing
1291 \def\filler@nss%
1292 {%
1293     \iffiller\ensuremath{\emptyset}\fi%
1294 }
```

`\c@nter@nss` Der folgende Makro setzt verschiedene Parameter, die zum Zentrieren eines Textes gebraucht werden.

```
1295 \def\c@nter@nss%
1296 {%
1297     \leftskip=0pt plus 2em \rightskip=\leftskip \parfillskip=0pt
1298     \spaceskip=.333em \xspaceskip=.5em
1299     \pretolerance=9999 \tolerance=9999
1300     \hyphenpenalty=9999 \exhyphenpenalty=9999%
1301 }
```

`\PositionNSS` Struktogramme werden intern als `minipage` verwaltet. Das Kommando `\PositionNSS` ermöglicht die Positionierung, es erwartet die Werte, die die `minipage`-Umgebung akzeptiert: `t`, `b` und keine Parameterangabe. Die Parameter werden nicht überprüft und gelten global ab Aufruf des Kommandos.

```
1302 \def\Position@nss{}%
1303 \newcommand{\PositionNSS}[1]%
1304 {%
1305     \def\Position@nss{#1}%
1306 }
```

`\struktogramm` `\struktogramm` beginnt ein Struktogramm

```
1307 \def\struktogramm(#1,#2)%
1308 {%
```

Zunächst werden die internen Kommandos aktiviert:

```
1309     \let\ifthenelse=\ifthenelse@nss%
1310     \let\ifend=\ifend@nss%
```

Falls die Option `draft` gesetzt wurde, wird für jedes Struktogramm grundsätzlich auf den Entwurfsmodus umgeschaltet.

```
1311     \ifdraft@nss\let\struktogramm@nss=\struktogramm@Proof\fi%
```

Um den optionalen Parameter zu behandeln, wird in Abhängigkeit vom nächsten Zeichen ein internes Makro aufgerufen.

```
1312     \@ifnextchar [{\struktogramm@nss(#1,#2)}{\struktogramm@nss(#1,#2)[]}%
1313 }
```

`\endstruktogramm` `\endstruktogramm` beendet das Struktogramm. Offene Gruppen und Umgebungen werden geschlossen, lokale Definitionen globaler Werte werden rückgängig gemacht.

```
1314 \def\endstruktogramm%
1315 {%
1316     \endstr@kt%
1317     \end{picture}%
1318 \end{minipage}%
1319 \setlength{\unitlength}{\savelength@nss}%
1320 }
```

`\struktogramm@NoProof` `\struktogramm@NoProof` öffnet ein Struktogramm

```
1321 \def\struktogramm@NoProof(#1,#2)[#3]%
1322 {%
    Zunächst wird die aktuelle \unitlength zwischengespeichert, da mit einer eigenen
    Größe gearbeitet wird.
```

```
1323 \def\next{#3}
1324 \setlength{\savelength@nss}{\unitlength}
1325 \setlength{\unitlength}{0.001mm}
```

Wenn der dritte Parameter nicht `\relax` ist, wird die Bezeichnung geschrieben und es muss ein bisschen zusätzlicher Platz geschaffen werden.

```
1326 \begin{minipage}[\Position@nss]{#1mm}
1327 \ifx\next\empty\else #3\[\smallskipamount]\fi%
```

Um die Verwaltung des benötigten Raumes möglichst einfach zu machen, wird eine `\picture`-Umgebung geöffnet, in der eine weitere Gruppe durch den Aufruf von `\str@kt` geöffnet wird.

```
1328 \begin{picture}(#1000,#2000)%
1329 \str@kt{0}{#2000}{#1000}%
1330 }
```

`\struktogramm@Proof` `\struktogramm@proof` öffnet ein Struktogramm im *proof*-Modus. Der Ablauf entspricht im wesentlichen `\struktogramm@NoProof`.

```
1331 \def\struktogramm@Proof(#1,#2)[#3]%
1332 {%
1333     \def\next{#3}%
1334     \setlength{\savelength@nss}{\unitlength}%
1335     \setlength{\unitlength}{0.001mm}%
1336     \begin{minipage}[\Position@nss]{#1mm}%
1337         \ifx\next\empty\else #3\[\smallskipamount]\fi%
1338     \begin{picture}(#1000,#2000)%
1339         \put(0,0){\makebox(0,0)[cc]{$\bullet$}}%
1340         \put(0,#2000){\makebox(0,0)[cc]{$\bullet$}}%
1341         \put(#1000,0){\makebox(0,0)[cc]{$\bullet$}}%
1342         \put(#1000,#2000){\makebox(0,0)[cc]{$\bullet$}}%
1343     \str@kt{0}{#2000}{#1000}%
1344 }
```

`\sProofOn` Die Schalter zum Umschalten zwischen *proof*- und *non proof*-Modus. Abschließend `\sProofOff` wird in den *non proof*-Modus umgeschaltet.

```
1345 \def\sProofOn%
1346 {%
```

```

1347 \let\struktogramm@nss=\struktogramm@Proof%
1348 }
1349 \def\sProofOff%
1350 {%
1351 \let\struktogramm@nss=\struktogramm@NoProof%
1352 }
1353 \let\struktogramm@nss=\struktogramm@NoProof%

```

Um Kompatibilität mit älteren Versionen von Struktogrammen zu erhalten, sind die Makros `\openstrukt` und `\closestrukt` weiterhin in diesem *package* erhalten. Diese sollten aber nicht weiter benutzt werden.

`\openstrukt` `\openstrukt` beginnt ein Struktogramm

```

1354 \def\openstrukt#1#2%
1355 {%
    The actual \unitlength will be stored intermediately, since it is worked with an
    own entity.
1356 \setlength{\savelength@nss}{\unitlength}
1357 \setlength{\unitlength}{0.001mm}

```

Um die Verwaltung des benötigten Raumes möglichst einfach zu machen, wird eine `\picture`-Umgebung geöffnet, in der eine eigene weitere Gruppe durch den Aufruf von `\str@kt` geöffnet wird.

```

1358 \begin{picture}(\#1000,\#2000)
1359 \str@kt{0}{\#2000}{\#1000}%
1360 }

```

Es ist zu beachten, dass es kein optionales Argument gibt.

`\closestrukt` `\closestrukt` beendet das Struktogramm. Offene Gruppen und Umgebungen werden geschlossen, lokale Definitionen globaler Werte werden rückgängig gemacht.

```

1361 \def\closestrukt%
1362 {%
1363 \endstr@kt
1364 \end{picture}
1365 \setlength{\unitlength}{\savelength@nss}%
1366 }

```

`\getoption` `\getoption` überprüft, ob ein optionales Argument angegeben wurde. Dies geschieht, indem das nächste Zeichen getestet wird.

```

1367 \def\getoption{ \@ifnextchar [{\getnum}{\param@nss=0\next}}

```

`\getnum` `\getnum` weist den Parameter dem globalen Zähler `\param@nss` zu und ruft den auf `\next` gesetzten Makro auf.

```

1368 \def\getnum[#1]{\param@nss=#1\next}

```

`\str@kt` `\str@kt` setzt globale Werte: `\gx@nss` als x-Koordinate (`#1`), `\gy@nss` als y-Koordinate (`#2`) und `\gsize@nss` als Breite des aktuellen Struktogramms (`#3`). Gleichzeitig wird eine Gruppe geöffnet, in der mit lokalen Kopien dieser Werte (`\x@nss`, `\y@nss` und `\xsize@nss`) gearbeitet werden kann.

```

1369 \def\str@kt#1#2#3%
1370 {%
1371 \global\x@nss=#1\global\gy@nss=#2\global\gsize@nss=#3%

```

```

1372   \begingroup%
1373     \x@nss=\gx@nss\y@nss=\gy@nss\ysize@nss=\gsize@nss%
1374 }

```

`\endstr@kt` schließt die oben geöffnete Gruppe, die aktuelle y -Koordinate wird global umge-
setzt, um den (weiter unten liegenden) Ansatzpunkt für das nächste Element des
Struktogramms zu erhalten.

```

1375 \def\endstr@kt%
1376 {%
1377     \global\gy@nss=\y@nss%
1378   \endgroup%
1379   \ydepth@nss=\gy@nss%
1380 }

```

`\set@nss` `\set@nss` setzt den als ersten Parameter übergebenen Text in eine horizontale
Box `\TextBox@nss`. Ist die Breite dieser Box größer als die im ersten Parameter
übergebene Breite, wird der Text in eine vertikale Box gesetzt.

```

1381 \def\set@nss#1#2%
1382 {%
1383   \setbox\TextBox@nss=\hbox{#1}%
1384   \tempxx@nss=\ysize@nss\advance\tempxx@nss by-#2%
1385   \ifdim\wd\TextBox@nss>\tempxx@nss\unitlength%
1386     \setbox\TextBox@nss=\vbox{\hsize=\tempxx@nss\unitlength\noindent#1}%
1387   \fi%
1388 }

```

`\block@nss` `\block@nss` setzt die Box `\TextBox@nss` entsprechend dem zweiten Parameter.
Ist dieser 1, wird eine Box mit Rahmen gesetzt, ist dieser 0, wird eine Box ohne
Rahmen gesetzt. Bei allen anderen Werten wird nur die Größe des Struktogramms,
gegeben durch `\y@nss`, vergrößert.

```

1389 \def\block@nss#1#2%
1390 {%
1391   \tempy@nss=#1\TextBoxHt@nss=\ht\TextBox@nss%
1392   \advance\TextBoxHt@nss by\dp\TextBox@nss%
1393   \advance\TextBoxHt@nss by 2mm%
1394   \ifdim\TextBoxHt@nss<\tempy@nss\unitlength%
1395     \TextBoxHt@nss=\tempy@nss\unitlength%
1396   \fi%
1397   \TextBoxHt@nss=0.00536\TextBoxHt@nss%
1398   \ydepth@nss=\TextBoxHt@nss\advance\y@nss by-\ydepth@nss%
1399   \ifx#21%
1400     \put(\x@nss,\y@nss)%
1401     {%
1402       \framebox(\ysize@nss,\ydepth@nss){\box\TextBox@nss}%
1403     }%
1404   \else%
1405     \ifx#20%
1406       \put(\x@nss,\y@nss)%
1407       {%
1408         \makebox(\ysize@nss,\ydepth@nss){\box\TextBox@nss}%
1409       }%
1410     \else%

```

```

1411         \advance\y@nss by \ydepth@nss%
1412     \fi%
1413 \fi%
1414 }

```

`\declarationtitle` Der Makro `\declarationtitle` legt den Inhalt der optionalen Überschrift bei Deklarationen fest. Er kann global vorgegeben werden, um eine einheitliche Darstellung zu erhalten.

```

1415 \def\declarationtitle{\ifnum\language=\languageNGerman Speicherplatz bereitstellen:
1416     \else providing memory space\fi}

```

`\descriptionindent` Die folgenden Variablen (*dimen-register*) wurden eingeführt, um das Layout der Variablenbeschreibung frei gestalten zu können.

```

\descriptionwidth@nss 1417 \newlength{\descriptionindent} \setlength{\descriptionindent}{1.5em}
\descriptionsep 1418 \newlength{\descriptionwidth} \setlength{\descriptionwidth}{40pt}
1419 \newlength{\descriptionsep} \setlength{\descriptionsep}{\tabcolsep}
1420 \newlength{\descriptionwidth@nss}

```

`\descriptionlabel@nss` Internes Kommando nach [GMS94, Abs. 3.3.5] zur Definition der Marke bei der Liste der Variablenbeschreibungen

```

1421 \newcommand{\descriptionlabel@nss}[1]%
1422 {%
1423     \settowidth{\descriptionwidth@nss}{#1}%
1424     \ifnum\descriptionwidth@nss>\descriptionwidth% % term > labelwidth
1425         \parbox[b]{\descriptionwidth}%
1426         {%
1427             \makebox[0pt][l]{#1}\%
1428         }%
1429     \else% % term < labelwidth
1430         #1%
1431     \fi%
1432     \hfil\relax%
1433 }

```

`declaration` Die Beschreibung von Variablen basiert nun auf der allgemeinen `list`-Umgebung, die entsprechend modifiziert wurde. Insbesondere wurden alle Abstände auf `Opt` gesetzt.

```

1434 \newenvironment{declaration}[1][\declarationtitle]%
1435 {%
1436     \def\next{#1}%
1437     \ifx\next\empty\else #1\\fi
1438     \let\description=\description@nss
1439     \tempxx@nss=\xsize@nss\advance\tempxx@nss by -\tempx@nss%
1440     \begin{minipage}[t]{\tempxx@nss\unitlength}
1441     \begin{list}{}%
1442     {%
1443         \renewcommand{\makelabel}{\descriptionlabel@nss}%
1444         \setlength{\labelwidth}{\descriptionwidth}%
1445         \setlength{\itemsep}{0pt} \setlength{\topsep}{0pt}%
1446         \setlength{\parsep}{0pt} \setlength{\partopsep}{0pt}
1447         \setlength{\leftmargin}{\descriptionwidth}%
1448         \addtolength{\leftmargin}{\descriptionsep}%
1449         \addtolength{\leftmargin}{\descriptionindent}%

```

```

1450 }%
1451 }%
1452 {%
1453 \end{list}%
1454 \end{minipage}
1455 }

```

`\description@nss` Um keine Schwierigkeiten mit der `description`-Umgebung zu haben, wird das Kommando `\description@nss` mit der Erweiterung `@nss` eingeführt. Seine Gültigkeit wird mit einem `\let` auf die `declaration`-Umgebung beschränkt

```

1456 \newcommand{\description@nss}[2]{\item[#1] \{#2\}}

```

`\assign` Einfache Anweisungen werden über `\assign` geschrieben:

```

1457 \def\assign{\let\next=\@assign\getoption}%
1458 \def\@assign#1%
1459 {%
1460 \temp@nss=\xsize@nss\divide\temp@nss by 6%
1461 \set@nss{#1}{\temp@nss}\temp@nss=\number\param@nss000%
1462 \block@nss{\temp@nss}{1}%
1463 }

```

`\sub` Der Aufruf eines Unterprogrammes wird mit `\sub` gekennzeichnet:

```

1464 \def\sub{\let\next=\@sub\getoption}
1465 \def\@sub#1%
1466 {%
1467 \@assign{#1}\temp@nss=\xsize@nss\divide\temp@nss by 20%
1468 \tempxx@nss=\x@nss\advance\tempxx@nss by \temp@nss%
1469 \put(\tempxx@nss,\y@nss){\line(0,1){\ydepth@nss}}%
1470 \tempxx@nss=\x@nss\advance\tempxx@nss by \xsize@nss%
1471 \advance\tempxx@nss by -\temp@nss%
1472 \put(\tempxx@nss,\y@nss){\line(0,1){\ydepth@nss}}%
1473 }

```

`\exit` Verlassen einer Schleife an einer bestimmten Stelle.

```

1474 \def\exit{\let\next=\@exit\getoption}
1475 \def\@exit#1%
1476 {%
1477 \temp@nss=\xsize@nss\divide\temp@nss by 6%
1478 \set@nss{#1}{\temp@nss}%
1479 \temp@nss=\number\param@nss000%
1480 \block@nss{\temp@nss}{1}%
1481 \divide\ydepth@nss by 2%
1482 \advance\y@nss by \ydepth@nss%
1483 \ifnum\ydepth@nss<3500%
    Eigene Gruppe bilden, um Änderung an \ydepth@nss lokal zu halten
1484 {%
1485 \multiply\ydepth@nss by 2%
1486 \put(\x@nss,\y@nss){\line(2,1){\ydepth@nss}}%
1487 \put(\x@nss,\y@nss){\line(2,-1){\ydepth@nss}}%
1488 }%
1489 \else%
1490 \put(\x@nss,\y@nss){\line(1,1){\ydepth@nss}}%
1491 \put(\x@nss,\y@nss){\line(1,-1){\ydepth@nss}}%

```

```

1492 \fi%
1493 \advance\y@nss by-\ydepth@nss%
1494 }

```

`\return` Die Rückkehr aus einem Unterprogramm wird mit `\return` gekennzeichnet:

```

1495 \def\return{\let\next=\@return\getoption}
1496 \def\@return#1%
1497 {%
1498   \tempy@nss=\number\param@nss000%
1499   \ifnum\tempy@nss<7000 \tempy@nss=7000 \fi%
1500   \set@nss{#1}{\tempy@nss} \block@nss{\tempy@nss}{1}%
1501   \divide\ydepth@nss by 2%
1502   \advance\y@nss by \ydepth@nss%
1503   \put(\x@nss,\y@nss){\line(1,1){\ydepth@nss}}%
1504   \put(\x@nss,\y@nss){\line(1,-1){\ydepth@nss}}%
1505   \tempx@nss=\x@nss\advance\tempx@nss by \xsize@nss%
1506   \put(\tempx@nss,\y@nss){\line(-1,1){\ydepth@nss}}%
1507   \put(\tempx@nss,\y@nss){\line(-1,-1){\ydepth@nss}}%
1508   \advance\y@nss by-\ydepth@nss%
1509 }

```

`\condindent@nss` `\index@nss`, `\anzzeilen@nss` und `\condindent@nss` werden benötigt, um den Umbruch der Bedingung in einer `\if`-Abfrage geeignet durchführen zu können:

<code>\index@nss</code>	ein Schleifenindex
<code>\anzzeilen@nss</code>	die Anzahl von Zeilen, in die der Bedingungstext maximal zerlegt wird
<code>\condindent@nss</code>	enthält den jeweiligen Wert des Einzugs, um den Bedingungstext an das Dreieck anzupassen

Die Höhe der Zeilen wird bei der Berechnung mit 5mm angenommen.

```

1510 \countdef\condindent@nss=232 \countdef\index@nss=233%

```

`\gindhelp@nss` Um die Textform (`\parshape`) an beliebige Steigungen anpassen zu können, sind einige Rechenoperationen mit Hilfsgrößen erforderlich. Die folgenden `dimen`-Variablen wurden eingeführt, um die Textform (`\parshape`) an beliebige Steigungen anpassen zu können, die Bedeutungen sind im einzelnen:

```

\indentmeasure@nss
\indentmeasureleft@nss
\thisindent@nss
\thisindentleft@nss
\thislength@nss

```

<code>\indentmeasure@nss:</code>	Einheit des Einzugs (links)
<code>\indentmeasureright@nss:</code>	Einheit des Einzugs (rechts)
<code>\thisindent@nss:</code>	aktueller Einzug (links)
<code>\thisindentright@nss:</code>	aktueller Einzug (rechts)
<code>\thislength@nss:</code>	Länge der aktuellen Zeile, ergibt sich aus der Zeilenlänge <code>\xsize@nss</code> vermindert um <code>\thisindent@nss</code> und <code>\thisindentright@nss</code>

`\gindhelp@nss` ist eine einfache Hilfsvariable zur temporären Speicherung numerischer Werte.

```

1511 \newdimen\indentmeasure@nss \newdimen\indentmeasureright@nss
1512 \newdimen\thisindent@nss \newdimen\thisindentright@nss
1513 \newdimen\thislength@nss
1514 \countdef\gindhelp@nss=234

```

`\ifthenelse@nss` `\ifthenelse@nss` leitet eine einfache Verzweigung ein. Um Konflikte mit anderen Makropaketen zu vermeiden, wird eine Schreibweise benutzt, die auf Grund des Klammeraffens nicht direkt aufrufbar ist. Erst ein `\let` in der Definition der Umgebung `struktogramm` macht dieses Kommando aktiv.

```
1515 \def\ifthenelse@nss{\let\next=\@ifthenelse@nss\getoption}
1516 \def\@ifthenelse@nss#1#2#3#4#5%
```

Die 5 Parameter sind:

1. Die Steigung (negativ, invers) des linken Zweiges,
2. die Steigung (positiv, invers) des rechten Zweiges,
3. die Bedingung, die getestet wird,
4. Fall des linken Zweiges (üblicherweise „ja“) und
5. Fall des rechten Zweiges (üblicherweise „nein“).

```
1517 {%
```

Als erstes werden die schrägen Linien gezeichnet. `\tempx@nss` ist die Breite des linken Zweiges, `\tempxx@nss` die des rechten Zweiges.

```
1518   %\ifthenelse{boolean{anygradient}}%
1519   {%
1520   \ifanygradient
```

Wenn `\anygradient` nicht wahr ist, werden schräge Linien mit `\line` erzeugt. `\xx@nss` ist die Anzahl von Einheiten, in die aktuelle Breite zerlegt wird.

```
1521   \xx@nss=#1 \advance\xx@nss by #2%
1522   \ifnum\number\param@nss000>0%
```

`\tempyy@nss` ist die Höhe des „Bedingungsrechtecks“, `\tempx@nss` die linke und `\tempxx@nss` die rechte Breite.

```
1523   \tempyy@nss=\number\param@nss000%
1524   \tempx@nss=\xsize@nss%
1525   \multiply\tempx@nss by #1 \divide\tempx@nss by \xx@nss%
1526   \tempxx@nss=\xsize@nss%
1527   \multiply\tempxx@nss by #2 \divide\tempxx@nss by \xx@nss%
1528   \else%
1529   \tempyy@nss=\xsize@nss \divide\tempyy@nss by \xx@nss%
1530   \tempx@nss=\tempyy@nss \multiply\tempx@nss by #1%
1531   \tempxx@nss=\tempyy@nss \multiply\tempxx@nss by #2%
1532   \fi%
1533   \xx@nss=\tempx@nss \advance\xx@nss by \x@nss%
1534   \tempy@nss=\y@nss \advance\tempy@nss by -\tempyy@nss%
1535   \ifelines%
1536   \emline{\x@nss}{\y@nss}{\xx@nss}{\tempy@nss}%
1537   \else\ifcurves%
1538   \curve(\x@nss, \y@nss, \xx@nss, \tempy@nss)%
1539   \else\ifpictIle%
```

Die Steigungen dürfen jetzt fast beliebige Werte annehmen, Zähler und Nenner müssen jedoch im Intervall $[-1000, 1000]$ liegen, die Skalierung mit dem Faktor 1000 muss daher wieder rückgängig gemacht werden.

```
1540   \tempyyy@nss=\tempyy@nss
```

```

1541         \divide\tempyyy@nss by 1000
1542         \tempxxx@nss=\tempx@nss
1543         \divide\tempxxx@nss by 1000
1544         \put(\x@nss,\y@nss){\line(\tempxxx@nss, -\tempyyy@nss){\tempx@nss}}
1545     \fi\fi\fi%
1546     \advance\x@nss by \xsize@nss%
1547     \ifemlines%
1548         \emline{\x@nss}{\y@nss}{\xx@nss}{\tempy@nss}{}%
1549     \else\ifcurves%
1550         \curve(\x@nss, \y@nss, \xx@nss, \tempy@nss)
1551     \else\ifpictIIE%
1552         \tempxxx@nss=\tempxx@nss
1553         \divide\tempxxx@nss by 1000
1554         \put(\x@nss,\y@nss){\line(-\tempxxx@nss, -\tempyyy@nss){\tempxx@nss}}
1555     \fi\fi\fi%

```

Die Koordinaten werden auf die linke untere Ecke des „Bedingungsrechteckes“ gesetzt:

```

1556     \advance\x@nss by -\xsize@nss \advance\y@nss by -\tempy@nss%

```

Nun wird die Form des Bedingungstextes berechnet (vgl. [Knu86, Kap. 14, S. 101–102]):

```

1557     \tempy@nss=\tempy@nss\divide\tempy@nss by 5000%

```

Wenn die Höhe des „Bedingungsrechteckes“ zu gering ist, muss dennoch eine Zeile hineingeschrieben werden, ansonsten führt \parshape zu einem Fehler.

```

1558     \ifnum\tempy@nss<1 \tempy@nss=1\fi%
1559     \indentmeasure@nss=\xsize@nss\unitlength%
1560     \gindhelph@nss=#1\advance\gindhelph@nss by #2%
1561     \divide\indentmeasure@nss by \gindhelph@nss%
1562     \indentmeasureright@nss = \indentmeasure@nss%
1563     \multiply\indentmeasure@nss by #1%
1564     \multiply\indentmeasureright@nss by #2%
1565     \divide\indentmeasure@nss by \tempyy@nss%
1566     \divide\indentmeasureright@nss by \tempyy@nss%
1567     \phantom{\vbox{\hsize=0pt\global\gindhelph@nss=\the\baselineskip}}}%
1568     \multiply\gindhelph@nss by 360%
1569     \multiply\indentmeasure@nss by \gindhelph@nss%
1570     \multiply\indentmeasureright@nss by \gindhelph@nss%
1571     \xdef\shape{\index@nss=0%
1572 \loop%
1573     \advance\index@nss by 1% %Zahl der Zeile
1574     \thisindent@nss=\indentmeasure@nss%
1575     \multiply\thisindent@nss by \index@nss%
1576     \advance\thisindent@nss by 0.175cm%
1577     \xdef\shape{\shape\the\thisindent@nss}%
1578     \thisindentright@nss=\indentmeasureright@nss%
1579     \multiply\thisindentright@nss by \index@nss%
1580     \advance\thisindentright@nss by 0.175cm%
1581     \thislength@nss=\xsize@nss\unitlength%
1582     \advance\thislength@nss by -\thisindent@nss
1583     \advance\thislength@nss by -\thisindentright@nss
1584     \xdef\shape{\shape\the\thislength@nss}%
1585     \ifnum\index@nss<\tempy@nss%
1586 \repeat%

```

```

1587     }%
1588     {%
1589 \else
1590     \ifnum\number\param@nss000>0%
1591         \typeout{struktex warning:}%
1592         \typeout{[...] not supplied without curves, emlines2, or pict2e}%
1593     \fi%
1594     \tempx@nss=#1 \advance\tempx@nss by #2%
1595     \tempy@nss=\xsize@nss \divide\tempy@nss by \tempx@nss%
1596     \tempx@nss=\tempy@nss \multiply\tempx@nss by #1%
1597     \put(\x@nss,\y@nss){\line(#1,-1){\tempx@nss}}%
1598     \tempxx@nss=\tempy@nss \multiply\tempxx@nss by #2%
1599     \advance\x@nss by \xsize@nss%
1600     \put(\x@nss,\y@nss){\line(-#2,-1){\tempxx@nss}}%
1601     \advance\x@nss by -\xsize@nss \advance\y@nss by -\tempy@nss%
1602     \tempy@nss=\tempy@nss\divide\tempy@nss by 5000%
1603     \ifnum\tempy@nss<1 \tempy@nss=1\fi%
1604     \xdef\shape{}\index@nss=0%
1605     \loop%
1606         \advance\index@nss by 1%
1607         \condindent@nss=#1\multiply\condindent@nss by\index@nss%
1608         \multiply\condindent@nss by 5000%
1609         \xdef\shape{\shape\the\condindent@nss\unitlength}%
1610         \condindent@nss=#1\advance\condindent@nss by #2%
1611         \multiply\condindent@nss by\index@nss%
1612         \multiply\condindent@nss by -5000%
1613         \advance\condindent@nss by \xsize@nss%
1614         \xdef\shape{\shape\the\condindent@nss\unitlength}%
1615         \ifnum\index@nss<\tempy@nss%
1616     \repeat%
1617     }%
1618     \fi
1619     \put(\x@nss,\y@nss)%
1620     {%
1621         \framebox(\xsize@nss,\tempy@nss)[t1]%
1622         {%
1623             \vbox%
1624             {%
1625                 \hsize=\xsize@nss\unitlength%
1626                 \parshape=\tempy@nss \shape \c@nter@nss%
1627                                     %dies sorgt f"ur den Einzug
1628                 \noindent\vrule width0pt height12pt \relax #3%
1629                                     %hier wird der Text ausgegeben
1630             }%
1631         }%
1632     }%
1633     \divide\tempy@nss by 2%
1634     \xx@nss=\xsize@nss\advance\xx@nss by-\tempx@nss%
1635     \advance\xx@nss by-\tempxx@nss%
1636     \ifodd\xx@nss\advance\tempxx@nss by 1\fi%
1637     \divide\xx@nss by 2\advance\tempxx@nss by \xx@nss%
1638     \advance\tempx@nss by \xx@nss%
1639     \put(\x@nss,\y@nss){\makebox(\tempx@nss,\tempy@nss){#4}}%
1640     \xx@nss=\x@nss\advance\xx@nss by \tempx@nss%

```

```

1641     \put(\xx@nss,\y@nss){\makebox(\tempxx@nss,\tempyy@nss){#5}}%
1642     \str@kt{\x@nss}{\y@nss}{\tempx@nss}%
1643 }

\change   \change trennt die Zweige beim \ifthenelse@nss
1644 \def\change%
1645 {%
1646     \endstr@kt\str@kt{\xx@nss}{\y@nss}{\tempxx@nss}%
1647 }

\ifend@nss \ifend@nss beendet \ifthenelse@nss, wird in struktogramm intern gemacht.
1648 \def\ifend@nss%
1649 {%
1650     \global\gy@nss=\y@nss\endgroup\tempy@nss=\gy@nss%
1651     \ifnum\tempy@nss<\ydepth@nss%
1652         \y@nss=\tempy@nss\advance\ydepth@nss by-\tempy@nss%
1653         \put(\xx@nss,\y@nss)%
1654         {%
1655             \framebox(\tempx@nss,\ydepth@nss)%
1656             {%
1657                 \ifnum\ydepth@nss>5000\filler@nss\fi%
1658             }%
1659         }%
1660     \else%
1661         \ifnum\tempy@nss>\ydepth@nss%
1662             \y@nss=\ydepth@nss\advance\tempy@nss by-\ydepth@nss%
1663             \put(\xx@nss,\y@nss)%
1664             {%
1665                 \framebox(\tempxx@nss,\tempy@nss)%
1666                 {%
1667                     \ifnum\tempy@nss>5000\filler@nss\fi%
1668                 }%
1669             }%
1670     \else%
1671         \y@nss=\ydepth@nss%
1672     \fi%
1673 \fi%
1674 }

\forever   Beginn und Ende einer Endlosschleife.
\foreverend 1675 \def\forever{\let\next=@forever\getoption}
1676 \def\@forever{
1677     \tempx@nss=\xsize@nss
1678     \divide\tempx@nss by 6
1679     \set@nss{\mbox{\rule{0mm}{3ex}}}{\tempx@nss}
1680     \tempy@nss=\number\param@nss000
1681     \block@nss{\tempy@nss}{2}
1682     \advance\x@nss by \ydepth@nss
1683     \advance\y@nss by -\ydepth@nss
1684     \tempxx@nss=\xsize@nss
1685     \advance\tempxx@nss by -\ydepth@nss
1686     \tempy@nss=\ydepth@nss
1687     \str@kt{\x@nss}{\y@nss}{\tempxx@nss}
1688 }

```

```

1689 \def\foreverend{
1690     \endstr@kt
1691     \advance\x@nss by-\tempy@nss
1692     \advance\ydepth@nss by -\tempy@nss
1693     \tempy@nss=\ydepth@nss
1694     \ydepth@nss=-\ydepth@nss
1695     \advance\ydepth@nss by \y@nss
1696     \advance\ydepth@nss by \tempy@nss
1697     \y@nss=\tempy@nss
1698     \put(\x@nss,\y@nss){\framebox(\xsize@nss,\ydepth@nss){}}
1699     \put(\x@nss,\y@nss){\makebox(\xsize@nss,\tempy@nss){\box\TextBox@nss}}
1700 }

```

\dfr Aus Kompatibilitatsgrunden werden fur \forever und \foreverend noch die folgenden Abkurzungen bereitgestellt.

```

1701 \let\dfr\forever
1702 \let\dfrend\foreverend

```

\while Beginn und Ende einer kopfgesteuerten Schleife.

```

\whileend 1703 \def\while{\let\next=@while\getoption}
1704 \def@while#1{\tempx@nss=\xsize@nss\divide\tempx@nss by 6
1705     \set@nss{#1\hskip\xsize@nss\unitlength}{\tempx@nss}
1706     \tempy@nss=\number\param@nss000 \block@nss{\tempy@nss}{0}
1707     \tempy@nss=\y@nss\advance\y@nss by \ydepth@nss
1708     \tempx@nss=\x@nss\advance\tempx@nss by \ydepth@nss
1709     \tempxx@nss=\xsize@nss\advance\tempxx@nss by -\ydepth@nss
1710     \str@kt{\tempx@nss}{\tempy@nss}{\tempxx@nss} }
1711 \def\whileend{\endstr@kt\tempy@nss=\y@nss\advance\tempy@nss by-\ydepth@nss
1712     \y@nss=\ydepth@nss
1713     \put(\x@nss,\y@nss){\framebox(\xsize@nss,\tempy@nss){}} }

```

\until Beginn und Ende einer fugesteuerten Schleife.

```

\untilend 1714 \def\until{\let\next=@until\getoption}
1715 \def@until#1{\tempx@nss=\xsize@nss\divide\tempx@nss by 6
1716     \set@nss{#1\hskip\xsize@nss\unitlength}{\tempx@nss}
1717     \tempy@nss=\number\param@nss000 \block@nss{\tempy@nss}{2}
1718     \advance\x@nss by \ydepth@nss\tempxx@nss=\xsize@nss
1719     \advance\tempxx@nss by -\ydepth@nss \tempy@nss=\ydepth@nss
1720     \str@kt{\x@nss}{\y@nss}{\tempxx@nss} }
1721 \def\untilend{\endstr@kt\advance\x@nss by-\tempy@nss
1722     \advance\ydepth@nss by -\tempy@nss
1723     \tempy@nss=\ydepth@nss\ydepth@nss=-\ydepth@nss%
1724     \advance\ydepth@nss by \y@nss
1725     \y@nss=\tempy@nss
1726     \put(\x@nss,\y@nss){\framebox(\xsize@nss,\ydepth@nss){}}
1727     \put(\x@nss,\y@nss){\makebox(\xsize@nss,\tempy@nss)
1728     {\box\TextBox@nss}}}

```

\cases@nss \cases@nss wird benotigt, um die Anzahl der Falle im linken Zweig des \case-Konstruktes berechnen zu konnen. \cases@nss nimmt entweder die Anzahl der Falle oder die Anzahl der Falle -1 an.

`\case` `\case`, `\caseend` und `\switch` stellen eine Mehrfachverzweigung dar. Es sind derzeit noch die alten Versionen `\caseold`, `\caseoldend` und `\switchold` zusätzlich enthalten, diese sollten in einer späteren Version gestrichen werden.

Der Makro hat vier Parameter:

1. Die Steigung (negativ, invers) der schrägen Linie,
2. die Anzahl der Fälle,
3. den Text der Bedingung und
4. den des ersten Falles

Zusätzlich gibt es einen optionalen vorangestellten Parameter, der es im Falle der Benutzung des `emlines2.sty` ermöglicht, die schräge Linie mit einer beliebigen Steigung zu zeichnen. Wird dieser Parameter genutzt, hat der erste Parameter eine geänderte Bedeutung: ist der Parameter eine gerade Zahl, wird die Schräge gerade durchgezogen (kein *default*-Zweig), ansonsten wird der letzte Zweig als Standardfall angesehen und somit werden die Schrägen entsprechend gezeichnet: die ersten $n - 1$ Fälle werden im linken Zweig untergebracht, der n -te Fall wird im rechten Zweig abgehandelt, vgl. dazu auch die Abbildungen auf Seite 16 und 17.

```
1729 \def\case{\let\next=\@case\getoption}
1730 \def\@case#1#2#3#4%
1731 {
```

Merken der Anzahl der Fälle

```
1732 \tempxx@nss=\number#2
```

Zunächst die schräge Linie zeichnen

```
1733 \ifanygradient%
1734 %%
```

– wenn `\anygradient` nicht wahr ist, werden schräge Linien mit `\line` erzeugt, wobei die Höhe des Rahmens für die Fallunterscheidungen aus seiner Breite und der vorgegebenen Steigung berechnet wird –,

```
1735 \ifnum\number\param@nss000>0
```

Wenn die Höhe des Rahmens vorgegeben ist, nimm diese, ansonsten berechne sie wie oben.

```
1736 \tempyy@nss=\number\param@nss000%
1737 \ifodd\number#1
1738 \advance\tempxx@nss by -1
1739 \fi
1740 \else
1741 \tempyy@nss=\xsize@nss
1742 \divide\tempyy@nss by #1
1743 \fi
1744 \ifnum\number#2=\tempxx@nss
```

`\tempxx@nss` ist unverändert, es wird also die Diagonale (gerade Linie) gezeichnet. Die Berechnung der Koordinaten basiert auf:

`\x@nss`: linke obere x-Koordinate des Kastens
`\y@nss`: linke obere y-Koordinate des Kastens
`\xsize@nss`: Breite des Kastens
`\tempyy@nss`: Höhe des Kastens
`\tempxx@nss`: Anzahl der Fälle (evtl. reduziert um 1)

```

1745     \xx@nss=\x@nss \advance\xx@nss by \xsize@nss
1746     \tempy@nss=\y@nss \advance\tempy@nss by -\tempyy@nss
1747     \ifemlines%
1748         \emline{\x@nss}{\y@nss}{\xx@nss}{\tempy@nss}{}%
1749     \else\ifcurves%
1750         \curve{\x@nss, \y@nss, \xx@nss, \tempy@nss}%
1751     \else\ifpictIIE%
1752         \tempxxx@nss=\xx@nss

```

Die Argumente von `\line` für die Steigung müssen im Intervall $[-1000, 1000]$ liegen, daher wird hier der Faktor 1000 wieder wegdividiert. Abschließend wird die Länge der Strecke berechnet.

```

1753         \advance\tempxxx@nss by -\x@nss
1754         \divide\tempxxx@nss by 1000
1755         \tempyyy@nss=\tempy@nss
1756         \advance\tempyyy@nss by -\y@nss
1757         \divide\tempyyy@nss by 1000
1758         \advance\xx@nss by -\x@nss
1759         \put{\x@nss,\y@nss}{\line(\tempxxx@nss, \tempyyy@nss){\xx@nss}}
1760     \fi\fi\fi%
1761 \else
1762     \tempx@nss=\xsize@nss
1763     \multiply\tempx@nss by \tempxx@nss \divide\tempx@nss by #2
1764     \xx@nss=\x@nss \advance\xx@nss by \tempx@nss
1765     \tempy@nss=\y@nss \advance\tempy@nss by -\tempyy@nss
1766     \ifemlines%
1767         \emline{\x@nss}{\y@nss}{\xx@nss}{\tempy@nss}{}%
1768     \else\ifcurves%
1769         \curve{\x@nss, \y@nss, \xx@nss, \tempy@nss}%
1770     \else\ifpictIIE%

```

Die Berechnung der Länge muss lokal sein, darum wird hier eine eigene Gruppe eingeführt anstatt die letzte Subtraktion rückgängig zu machen.

```

1771         \begingroup
1772             \tempxxx@nss=\xx@nss
1773             \advance\tempxxx@nss by -\x@nss
1774             \divide\tempxxx@nss by 1000
1775             \tempyyy@nss=\tempy@nss
1776             \advance\tempyyy@nss by -\y@nss
1777             \divide\tempyyy@nss by 1000
1778             \advance\xx@nss by -\x@nss
1779             \put{\x@nss,\y@nss}{\line(\tempxxx@nss,
1780                 \tempyyy@nss){\xx@nss}}
1781         \endgroup
1782     \fi\fi\fi%
1783     \advance\x@nss by \xsize@nss
1784     \ifemlines%
1785         \emline{\x@nss}{\y@nss}{\xx@nss}{\tempy@nss}{}%
1786     \else\ifcurves%
1787         \curve{\x@nss, \y@nss, \xx@nss, \tempy@nss}%
1788     \else\ifpictIIE%
1789         \tempxxx@nss=\x@nss
1790         \advance\tempxxx@nss by -\xx@nss
1791         \divide\tempxxx@nss by 1000
1792         \tempyyy@nss=\tempy@nss

```

```

1793         \advance\tempyyy@nss by -\y@nss
1794         \divide\tempyyy@nss by 1000
1795         \xx@nss=-\xx@nss
1796         \advance\xx@nss by \x@nss
1797         \put(\x@nss,\y@nss){\line(-\tempxxx@nss,\tempyyy@nss){\xx@nss}}
1798     \fi\fi\fi%
1799     \advance\x@nss by -\xsize@nss
1800 \fi
1801 %}{
1802 \else% \ifanygradient
1803     \ifnum\number\param@nss000>0
1804         \typeout{struktex warning:}%
1805         \typeout{[...] not supplied without curves, emlines2, or pict2e}%
1806     \fi
1807     \put(\x@nss,\y@nss){\line(#1,-1){\xsize@nss}}
1808     \tempyy@nss=\xsize@nss
1809     \divide\tempyy@nss by #1
1810 %}
1811 \fi% \ifanygradient

```

dann den Bedingungstext in die Mitte des rechten oberen Viertels des CASE-Rechtecks schreiben,

```

1812 \tempx@nss=\xsize@nss
1813 \divide\tempx@nss by 2
1814 \divide\tempyy@nss by 2
1815 \advance\x@nss by \tempx@nss
1816 \advance\y@nss by -\tempyy@nss
1817 \put(\x@nss,\y@nss){\makebox(\tempx@nss,\tempyy@nss){#3}}

```

schließlich $\x@nss$ und $\y@nss$ auf den alten Wert zuruecksetzen und zur späteren Verwendung merken.

```

1818 \advance\x@nss by -\tempx@nss
1819 \advance\y@nss by \tempyy@nss
1820 \xx@nss=\x@nss
1821 \yy@nss=\y@nss

```

$\tempx@nss$ ist die Breite eines Zweiges, die für alle Zweige gleich ist – Gesamtbreite geteilt durch Anzahl der Fälle –, $\tempy@nss$ die Höhe des Falltextes, die von Zweig zu Zweig kleiner wird, $\condindent@nss$ ist die Schrittweite, um die sich die Höhe des Falltextes bei jedem `\switch` verringert. Höhe und Schrittweite hängen davon ab, ob ein *default*-Zweig eingeführt wurde oder nicht, sie werden demzufolge durch $\tempxx@nss$ gesteuert.

```

1822 \tempx@nss=\xsize@nss
1823 \divide\tempx@nss by #2
1824 \ifnum\number\param@nss000>0
1825     \tempy@nss=\number\param@nss000%
1826 \else
1827     \tempy@nss=\xsize@nss
1828     \divide\tempy@nss by #1
1829 \fi
1830 \condindent@nss=\tempy@nss
1831 \divide \condindent@nss by \tempxx@nss

```

Links oben im linken Zweig den Text des Falles eintragen

```

1832 \advance\y@nss by -\tempy@nss

```

```

1833 \put(\x@nss,\y@nss)%
1834 {%
1835 \makebox(\temp@nss,\temp@nss)[lb]{\raise3pt\hbox{~#4}}%
1836 }%
1837 \ydepth@nss=1000000
1838 \str@kt{\x@nss}{\y@nss}{\temp@nss}
1839 }

```

Der Makro `\switch` für die einzelnen Fälle hat zwei Parameter:

1. einen optionalen Parameter, der angibt, wo der Text der Bedingung gesetzt wird: `l` oder `r` – der Standardwert ist `l` –, und
2. den Bedingungstext

```

1840 \def\switch%
1841 {%
1842 \@ifnextchar [{\@switch}{\@switch[1]}%
1843 }
1844 \def\@switch[#1]#2%
1845 {
1846 \global\gy@nss=\y@nss\endgroup
1847 \temp@nss=\gy@nss
1848 \ifnum\temp@nss<\ydepth@nss
1849 \ydepth@nss=\temp@nss
1850 \fi

```

Nun in den nächsten Zweig gehen und die Falltexthöhe vermindern.

```

1851 \advance\x@nss by \temp@nss
1852 \advance\temp@nss by -\condindent@nss
1853 \put(\x@nss,\y@nss)%
1854 {\makebox(\temp@nss,\temp@nss)[#1b]{\raise3pt\hbox{~#2~}}}
1855 \str@kt{\x@nss}{\y@nss}{\temp@nss}
1856 }
1857 \def\caseend%
1858 {
1859 \global\gy@nss=\y@nss\endgroup
1860 \temp@nss=\gy@nss
1861 \ifnum\temp@nss<\ydepth@nss
1862 \ydepth@nss=\temp@nss
1863 \fi
1864 \x@nss=\xx@nss
1865 \y@nss=\ydepth@nss
1866 \temp@nss=\yy@nss
1867 \advance\temp@nss by -\ydepth@nss
1868 \put(\x@nss,\y@nss){\framebox(\xsize@nss,\temp@nss){}}
1869 \temp@nss=\x@nss

```

Die folgende Schleife ist dafür zuständig, die Trennlinien zwischen den Falltexten zu ziehen.

```

1870 \loop
1871 \put(\temp@nss,\y@nss){\line(0,1){\temp@nss}}
1872 \advance\temp@nss by -\condindent@nss
1873 \advance\temp@nss by \temp@nss

```

Die Bedeutung der folgenden Zeilen ist noch unklar, was soll die 100?

```

1874     \xx@nss=\tempxx@nss
1875     \advance\xx@nss by -\x@nss
1876     \advance\xx@nss by 100
1877     \ifnum \xx@nss<\xsize@nss
1878     \repeat
1879 }

```

`\centerNss` Makro zur Zentrierung von Struktogrammen: da einfaches Zentrieren mit `\centering` nicht funktioniert, wird mit `\centerline` gearbeitet:

```

1880 \newbox\CenterBox@nss%
1881 \def\centernss%
1882 {%
1883     \begin{trivlist}%
1884     \item[] \strut%
1885     \setbox\CenterBox@nss=\hbox%
1886     \bgroup%
1887 }
1888 \def\endcenternss%
1889 {%
1890     \egroup%
1891     \strut\hfill\box\CenterBox@nss\hfill\strut%
1892     \end{trivlist}%
1893 }

```

`\CenterNssFile` Makro zur Zentrierung von Struktogrammen: da einfaches Zentrieren mit `\centering` nicht funktioniert, wird mit `\centerline` gearbeitet:

```

1894 \def\CenterNssFile#1%
1895 {%
1896     \begin{trivlist}%
1897     \item[] \setbox\CenterBox@nss=\hbox{\input{#1.nss}}%
1898     \strut\hfill\box\CenterBox@nss\hfill\strut%
1899     \end{trivlist}%
1900 }
1901 \let\centernssfile=\CenterNssFile

```

9.6 Zusätzliche Makros (Verifikation)

`\assert` `\assert` wird wie `\assign` definiert, führt ohne die Option `verification` aber `\assert@nss` nicht zu sichtbaren Ergebnissen:

```

1902 \newcommand\assert{\let\next=\assert@nss\getoption}%
1903 \newcommand\assert@nss[1]{%

```

Der folgende Code ist nur gültig, wenn die Option `verification` gesetzt wurde:

```

1904 \ifthenelse{\boolean{verification}}
1905 {

```

`\xsize@nss` Da `\oval` einen anderen Referenzpunkt als `\framebox` hat, brauchen wir zusätzliche Zähler, um den anderen Referenzpunkt ansprechen zu können.

```

1906 \countdef\xsize@nss=232%
1907 \countdef\ydepth@nss=233%

```

Für den Fall, dass die Option `verification` gesetzt wurde, wird der intern genutzte Makro `\assert@nss` neu definiert.

```

1908 \renewcommand{\assert@nss}[1]%
1909 {%
1910   \temp@nss=\xsize@nss\divide\temp@nss by 6%
1911   \set@nss{#1}{\temp@nss}\temp@nss=\number\param@nss000%
1912   \assertblock@nss{\temp@nss}{1}%
1913 }

```

`\assertblock@nss` `\assertblock@nss` setzt die Box `\TextBox@nss` entsprechend dem zweiten Parameter. Ist dieser 1, wird eine Box mit Rahmen gesetzt, ist dieser 0, wird eine Box ohne Rahmen gesetzt. Bei allen anderen Werten wird nur die Größe des Struktogramms, gegeben durch `\y@nss`, vergrößert.

```

1914 \newcommand{\assertblock@nss}[2]%
1915 {%
1916   \temp@nss=#1\TextBoxHt@nss=\ht\TextBox@nss%
1917   \advance\TextBoxHt@nss by\dp\TextBox@nss%
1918   \advance\TextBoxHt@nss by 2mm%
1919   \ifdim\TextBoxHt@nss<\temp@nss\unitlength%
1920     \TextBoxHt@nss=\temp@nss\unitlength%
1921   \fi%
1922   \TextBoxHt@nss=0.00536\TextBoxHt@nss%
1923   \ydepth@nss=\TextBoxHt@nss\advance\y@nss by-\ydepth@nss%
1924   \xsizeo@nss=\xsize@nss \temp@nss=\x@nss
1925   \divide\xsizeo@nss by 2%
1926   \advance\temp@nss by+\xsizeo@nss%
1927   \ydeptho@nss=\ydepth@nss \temp@nss=\y@nss
1928   \divide\ydeptho@nss by 2%
1929   \advance\temp@nss by+\ydeptho@nss%
1930   \ifx#21%
1931     \put(\x@nss,\y@nss)%
1932     {%
1933       \framebox(\xsize@nss,\ydepth@nss){\box\TextBox@nss}%
1934     }%
1935     \put(\temp@nss,\temp@nss){\oval(\xsize@nss,\ydepth@nss)}%
1936   \else%
1937     \ifx#20%
1938       \put(\x@nss,\y@nss)%
1939       {%
1940         \makebox(\xsize@nss,\ydepth@nss){\box\TextBox@nss}%
1941       }%
1942       \put(\temp@nss,\temp@nss){\oval(\xsize@nss,\ydeptho@nss)}%
1943     \else%
1944       \advance\y@nss by \ydepth@nss%
1945     \fi%
1946   \fi%
1947 }
1948 }
1949 {} % end of \ifthenelse{\boolean{verification}}

```

Literatur

- [Fut89] Gerald Futschek. *Programmentwicklung und Verifikation*. Springer Verlag, Wien – New York, 1989. 2, 43
- [GMS94] Michel Goossens, Frank Mittelbach and Alexander Samarin. *The L^AT_EX-Companion*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1994. 2, 54
- [GMS04] Frank Mittelbach and Michel Goossens. *The L^AT_EX-Companion*. Addison-Wesley Publishing Company, Reading, Massachusetts, second edition, 2004.
- [Knu86] D. E. Knuth. *The T_EX-Book*. Addison-Wesley, Reading, Massachusetts, 1986. 58
- [MDB94] Frank Mittelbach, Denys Duchier and Johannes Braams. *The DocStrip program*, Dezember 1994.
- [MDB01] Frank Mittelbach, Denys Duchier, Johannes Braams, Marcin Woliński and Mark Wooding *The DocStrip program*, September 2001. 4
- [Mit94] Frank Mittelbach. *The doc and shortvrb Packages*, Oktober 1994.
- [Mit01] Frank Mittelbach. *The doc and shortvrb Packages*, September 2001. 4
- [Neu96] Marion Neubauer. Feinheiten bei wissenschaftlichen Publikationen – Mikrotypographie-Regeln, Teil I. *Die T_EXnische Komödie*, 8(4):23–40, Februar 1996. 29
- [Rah92] Sebastian Rahtz. *The oz package*, 1992. 44

Index

Die kursiven Zahlen bezeichnen die Seite, auf der der jeweilige Eintrag beschrieben ist, unterstrichene Zahlen verweisen auf seine Definition; alle anderen Zahlen geben Stellen an, an denen der Eintrag auftritt.

	Symbols	<code>\@ifdefinable</code>	<code>\~</code> 568
<code>\"</code>	873 470, 478, 485	
<code>\#</code>	1177, 1249	<code>\@ifnextchar</code>	Numbers
<code>\&</code>	837, 1169, 1177	. 466, 468, 491,	<code>\0</code> 873
<code>\(</code> 42, 46, 57, 58, 61,		1312, 1367, 1842	<code>\sqcup</code> 500, 503, 738,
63–65, 67, 68,		<code>\@ifstar</code> 673	739, 832–837, 858
70–72, 75–77,		<code>\@ifthenelse@nss</code> . .	
92, 93, 97, 98,	 1515, 1516	A
100–104, 106–		<code>\@makeoether</code> 513, 552	<code>\Abb</code> 464
110, 112–116,		<code>\@namedef</code> . 475, 476,	<code>\active</code> . . . 513, 553, 568
119–123, 125,		481, 482, 486–488	<code>\addtolength</code>
143, 162, 173,		<code>\@nameuse</code> 486	. . . 398, 1448, 1449
184, 199, 218,		<code>\@newctr</code> 471	<code>\addvspace</code> . . . 521, 538
229, 240, 255,		<code>\@nexmpl</code> 466, 467	<code>\AlsoImplementation</code> 759
274, 285, 296,		<code>\@nil</code> 577,	<code>\anzeilen@nss</code> . . . <u>1510</u>
311, 330, 341, 352		581, 582, 587, 589	<code>\argi@nss</code> . . 1155, 1156
<code>\)</code> 42, 46, 57, 58, 61,		<code>\@nobreaktrue</code> . 499, 502	<code>\argii@nss</code> . 1155, 1159
63–65, 67, 68,		<code>\@nocounterr</code> 484	<code>\argiii@nss</code> . 1155, 1162
70–72, 75–77,		<code>\@oexmpl</code> 466, 483	<code>\assert</code> . 96, 98, 100,
92, 93, 97, 98,		<code>\@opargbeginexample</code>	104, 106, 108,
100–104, 106–	 495, 501	110, 112, 116,
110, 112–116,		<code>\@return</code> . . . 1495, 1496	119, 123, 125, <u>1902</u>
119–123, 125,		<code>\@setmcodes</code>	<code>\assert@nss</code> . <u>1902</u> , 1908
143, 162, 173,		. . . 1131, 1136,	<code>\assertblock@nss</code> . .
184, 199, 218,		1137, 1141, 1142 1912, <u>1914</u>
229, 240, 255,		<code>\@sub</code> 1464, 1465	<code>\assign</code> . . 42, 44, 46,
274, 285, 296,		<code>\@switch</code> . . . 1842, 1844	54, 63–65, 70–
311, 330, 341, 352		<code>\@tempa</code> 587, 589	72, 75–77, 89,
<code>\,</code> 459, 461		<code>\@tempc</code> 602, 615, 625,	101–103, 113–
<code>\@addtofilelist</code> . . . 653		627, 628, 631, 639	115, 120–122, <u>1457</u>
<code>\@assign</code> 1457, 1458, 1467		<code>\@temptokena</code>	<code>\AtBeginDocument</code> 777,
<code>\@beginexample</code> 493, 498		. 588, 595, 600,	778, 782, 783,
<code>\@bsphack</code> 511, 550		606, 613, 623, 637	787, 788, 792,
<code>\@case</code> 1729, 1730		<code>\@until</code> 1714, 1715	793, 798, 799,
<code>\@currentenv</code> 631, 633, 634		<code>\@verbatim</code> 676	802, 807, 808,
<code>\@definecounter</code> 471, 479		<code>\@vobeyspaces</code> 674	811, 819, 820, 823
<code>\@doendpe</code> 677		<code>\@warning</code> 644	
<code>\@endexample</code>		<code>\@while</code> 1703, 1704	B
. 476, 482, 488, 504		<code>\@xexmpl</code> 491, 492	<code>\bfseries</code> . 156, 159,
<code>\@esphack</code> 517, 560		<code>\@xnexmpl</code> 468, 469	167, 170, 178,
<code>\@exit</code> 1474, 1475		<code>\@yexmpl</code> 491, 494	181, 212, 215,
<code>\@exmpl</code> 475, 481, 487, 489		<code>\@ynexmpl</code> 468, 477	223, 226, 234,
<code>\@exmplcounter</code>		<code>\{</code> . . 456, 458, 567, 1456	237, 268, 271,
. 474, 480, 496		<code>\}</code> . . 456, 458, 567, 1456	279, 282, 290,
<code>\@exmplcountersep</code> .		<code>\^</code> 513,	293, 324, 327,
. 473, 497		553, 568, 1169, 1177	335, 338, 346,
<code>\@file@und</code> 677		<code>_</code> 1169, 1176	349, 500, 503, 1151
<code>\@forever</code> . . 1675, 1676		<code>\ </code> 26	

<code>\block@nss</code> 1389 , 1462 , 1480, 1500, 1681, 1706, 1717	1070, 1078, 1084, 1090, 1095	F <code>\fdesc@font</code> ... 371 , 376, 382, 385, 391
<code>\boolean</code> 1518 , 1904 , 1949	<code>\DeclareSymbolFont</code> 1114 , 1115	<code>\fdesc@label</code> .. 369 , 395
<code>\btt</code> 1121	<code>\DeclareSymbolFontAlphabet</code> 1116	<code>\fdesc@len</code> 368 , 371 , 373 , 379
C	<code>\description</code> 57, 58, 92, 93, 1438	<code>\filler@nss</code> 1289 , 1657 , 1667
<code>\c@enter@nss</code> . 1295 , 1626	<code>\description@nss</code> 1438 , 1456	<code>\fontencoding</code> 1128
<code>\case</code> 41 , 960 , 1729	<code>\descriptionindent</code> 1417 , 1449	<code>\fontfamily</code> 1128
<code>\caseend</code> 46 , 1729	<code>\descriptionlabel@nss</code> 1421 , 1443	<code>\fontseries</code> 1128
<code>\cases@nss</code> 1729	<code>\descriptionsep</code> 1417 , 1448	<code>\fontshape</code> 1129
<code>\Cee@nss</code> ... 1186 , 1216	<code>\descriptionwidth</code> 1417 , 1424 , 1425 , 1444 , 1447	<code>\foreign</code> 463
<code>\CenterBox@nss</code> 1880 , 1885 , 1891 , 1897 , 1898	<code>\descriptionwidth@nss</code> .. 1417 , 1423 , 1424	<code>\forever</code> 982 , 1675 , 1701
<code>\centerNss</code> 1880	<code>\dfr</code> 1701	<code>\foreverend</code> . 1675 , 1702
<code>\centernss</code> 1881	<code>\dfrend</code> 1701	<code>\frac</code> 42 , 46
<code>\CenterNssFile</code> ... 1894	<code>\dh</code> 462	<code>\futurelet</code> 580
<code>\centernssfile</code> ... 1901	<code>\dhspace</code> 461 , 462	G
<code>\change</code> .. 62 , 69 , 74 , 99 , 111 , 118 , 1644	<code>\do</code> 513 , 552	<code>\GetFileInfo</code> ... 15 , 877
<code>\char</code> 456 , 458	<code>\documentclass</code> .. 1 , 48 , 82 , 129 , 363 , 874	<code>\getnum</code> 1367 , 1368
<code>\check@percent</code> 666	<code>\dospecials</code> ... 513 , 552	<code>\getoption</code> 1367 , 1457 , 1464 , 1474 , 1495 , 1515 , 1675 , 1703 , 1714 , 1729 , 1902
<code>\CheckForKeyword@nss</code> ... 1187 , 1192 , 1199 , 1205 , 1209 , 1212 , 1223	E	<code>\gets</code> 42 , 46 , 63–65 , 70–72 , 75–77 , 101–103 , 113–115 , 120–122
<code>\closein</code> 658	<code>\edef</code> 633	<code>\gindhelp@nss</code> 1511 , 1560 , 1561 , 1567–1570
<code>\closeout</code> 517 , 559	<code>\emline</code> . 1536 , 1548 , 1748 , 1767 , 1785	<code>\gsize@nss</code> 1278 , 1371 , 1373
<code>\closestrukt</code> 1361	<code>\emptyset</code> .. 1290 , 1293	<code>\gx@nss</code> 1278 , 1371 , 1373
<code>\CodelineNumbered</code> . 757	<code>\encodingdefault</code> 1114 , 1128	<code>\gy@nss</code> 1278 , 1371 , 1373 , 1377 , 1379 , 1650 , 1846 , 1847 , 1859 , 1860
<code>\colortrue</code> 444	<code>\endcenterNss</code> 1880	H
<code>\complex</code> 1117	<code>\endcenternss</code> 1888	<code>\hexnumber@</code> . 1136 , 1137
<code>\condindent@nss</code> 1510 , 1607– 1614 , 1830 , 1831 , 1852 , 1872	<code>\endlinechar</code> .. 655 , 657	<code>\href</code> 446
<code>\count</code> 1131–1133	<code>\endList</code> ... 1226 , 1230	I
<code>\countdef</code> 1279–1284 , 1286 , 1510 , 1514 , 1906 , 1907	<code>\endlist</code> 504	<code>\if</code> 573 , 593 , 599 , 601 , 611 , 619 , 621 , 643
<code>\curve</code> .. 1538 , 1550 , 1750 , 1769 , 1787	<code>\endstr@kt</code> 1316 , 1363 , 1375 , 1646 , 1690 , 1711 , 1721	<code>\ifanygradient</code> 1520 , 1733 , 1802 , 1811
D	<code>\endstruktogramm</code> . 1314	<code>\ifcat</code> 546
<code>\declaration</code> 1434	<code>\endtrivlist</code> 677	<code>\ifcolor</code> 444 , 447
<code>\declarationtitle</code> 1415 , 1434	<code>\endverbatimwrite</code> . 558	
<code>\DeclareFontFamily</code> 1121	<code>\ensuremath</code> 1293	
<code>\DeclareFontShape</code> 1122	<code>\envb</code> 455	
<code>\DeclareMathSymbol</code> 1289	<code>\enve</code> 457	
<code>\DeclareOption</code> 1056 , 1063 ,	<code>\exec</code> .. 1225 , 1232 , 1235	
	<code>\exit</code> 1474	
	<code>\Expr</code> 422	

<code>\ifcurves</code>	1101, 1537, 1549, 1749, 1768, 1786		
<code>\ifdraft@nss</code> ..	1088, 1092, 1097, 1311		
<code>\ifemlines</code>	1103, 1535, 1547, 1747, 1766, 1784		
<code>\ifend</code>	66, 73, 78, 105, 117, 124, 1310		
<code>\ifend@nss</code>	1310, <u>1648</u>		
<code>\ifeof</code>	650, 664		
<code>\iffalse</code>	1088, 1093, 1097		
<code>\IfFileExists</code>	444, 676		
<code>\iffiller</code>	1293		
<code>\iffinal@nss</code> 1089, 1093, 1098		
<code>\ifpdf</code>	445		
<code>\ifpictIIe</code>	1105, 1285, 1539, 1551, 1751, 1770, 1788		
<code>\ifthenelse</code>	61, 67, 68, 97, 107, 109, 993, 1309, 1518, 1904, 1949		
<code>\ifthenelse@nss</code> 1309, <u>1515</u>		
<code>\iftrue</code>	1089, 1092, 1098		
<code>\immediate</code>	512, 515, 517, 551, 555, 559		
<code>\indentmeasure@nss</code>	<u>1511</u> , 1559, 1561–1563, 1565, 1569, 1574		
<code>\indentmeasureleft@nss</code>	<u>1511</u>		
<code>\indentmeasureright@nss</code>	1511, 1562, 1564, 1566, 1570, 1578		
<code>\index</code>	412		
<code>\index@nss</code>	<u>1510</u> , 1571, 1573, 1575, 1579, 1585, 1604, 1606, 1607, 1611, 1615		
<code>\integer</code>	<u>1117</u>		
<code>\itemsep</code>	1445		
<code>\itshape</code>	156, 159, 178, 181, 212, 215, 234, 237, 268, 271, 290, 293, 324, 327, 346, 349		
		J	1596, 1598, 1607, 1608, 1611, 1612, 1763
		<code>\Java@nss</code> ..	1191, <u>1216</u>
		<code>\jobname</code> ..	512, 529, 536
		L	
		<code>\labelsep</code>	398
		<code>\labelwidth</code>	379, 380, 396, 397, 1444
		<code>\language</code>	506, 1415
		<code>\languageNGerman</code>	32, 506, 1415
		<code>\LaTeX@nss</code>	1208, <u>1216</u>
		<code>\lccode</code>	567–569
		<code>\ldots</code>	435
		<code>\le</code>	61, 67, 68, 97, 98, 106–110, 112, 116, 119, 123, 125
		<code>\leftmargin</code>	397, 398, 499, 502, 1447–1449
		<code>\line</code>	1469, 1472, 1486, 1487, 1490, 1491, 1503, 1504, 1506, 1507, 1544, 1554, 1597, 1600, 1759, 1779, 1797, 1807, 1871
		<code>\linewidth</code> 523, 524, 533, 534
		<code>\loop</code>	1132, 1572, 1605, 1870
		<code>\lower</code>	1109, 1110
		<code>\lowercase</code>	570
		M	
		<code>\m@ne</code>	655
		<code>\MacroFont</code>	672
		<code>\mathbb</code>	1116–1120
		<code>\mathcode</code>	1132
		<code>\MathItalics</code>	<u>1131</u>
		<code>\MathNormal</code>	<u>1131</u>
		<code>\mathord</code>	1289
		<code>\mbox</code>	500, 503, 1231, 1232, 1252, 1253, 1257, 1258, 1679
		<code>\meta</code>	454
		<code>\multiply</code>	1485, 1525, 1527, 1530, 1531, 1563, 1564, 1568–1570, 1575, 1579,
		N	
		<code>\nat</code>	<u>1117</u>
		<code>\newboolean</code> ..	1052– 1055, 1077, 1082
		<code>\newcommand</code>	369, 459, 461, 464, 1150– 1153, 1166, 1173, 1183, 1216, 1261– 1265, 1303, 1421, 1456, 1902, 1903, 1914
		<code>\newenvironment</code> 389, 509, 1434
		<code>\newexample</code>	465, 506
		<code>\newlength</code>	368, 1417–1420
		<code>\newread</code>	646
		<code>\newtoks</code>	541
		<code>\newwrite</code>	508
		<code>\noexpand</code>	473, 496, 573, 593, 599, 601, 611, 619, 621, 633, 634, 643
		<code>\nofiles</code>	133, 875
		O	
		<code>\obeylines</code>	140, 150, 158, 169, 180, 196, 206, 214, 225, 236, 252, 262, 270, 281, 292, 308, 318, 326, 337, 348
		<code>\openin</code>	649
		<code>\openout</code>	512, 551
		<code>\openstrukt</code>	<u>1354</u>
		<code>\OSII@nss</code>	1222
		<code>\oval</code>	1935, 1942
		P	
		<code>\pageref</code>	451
		<code>\paragraph</code> 147, 155, 166, 177, 203, 211, 222, 233, 259, 267, 278, 289, 315, 323, 334, 345
		<code>\param@nss</code>	<u>1279</u> , 1367, 1368, 1461, 1479,

1498,	1522,	<code>\pLanguage</code>	137, 192,	274, 283, 285,
1523,	1590,		248, 304, 407, 1216	294, 296, 301,
1680,	1706,	<code>\pLanguage@nss</code>	1186,	309, 311, 328,
1717,	1735,		1191, 1198,	330, 339, 341,
1736,	1803,		1204, 1208, 1216	350, 352, 357, 1173
1824,	1825, 1911	<code>\Position@nss</code>	. 1302,	<code>\pVariable@</code>
<code>\parsep</code> 1446		1305, 1326, 1336	. 1178, 1183
<code>\partopsep</code> 1446	<code>\PositionNSS</code> 1302	<code>\pVariable</code>
<code>\Pascal@nss</code>	. 1198, 1216	<code>\pparg</code> 453	141, 160,
<code>\pBoolValue</code>	148, 151,	<code>\ProcessOptions</code>	.. 1100	171, 182, 197,
	193, 204, 207,	<code>\protect</code> 450, 451	216, 227, 238,
	249, 260, 263,	<code>\providecommand</code>	...	253, 272, 283,
	305, 316, 319, 1265	 453, 455, 457	294, 309, 328,
<code>\pComment</code> 1242	<code>\ProvidesFile</code> 654	339, 350, 1173 , 1271
<code>\pCommentFont@nss</code>	.	<code>\ProvidesPackage</code>	.. 442	<code>\Python@nss</code>
	.. 1150 , 1163, 1245	<code>\pTrue</code>	144, 152, 163,	. 1204, 1216
<code>\pExp</code> 432, 1166		174, 185, 200,	R
<code>\pExpFont@nss</code>	. 1150 ,		208, 219, 230,	<code>\read</code>
	1157, 1225, 1258		241, 256, 264, 663
<code>\pExpression</code> 1166		275, 286, 297,	<code>\real</code>
<code>\pFalse</code>	144, 152, 163,		312, 320, 331, 1117
	174, 185, 200,		342, 353, 1263 , 1273	<code>\ref</code>
	208, 219, 230,	<code>\pTrue@nss</code> 450, 464
	241, 256, 264,		.. 1261 , 1263, 1267	<code>\refstepcounter</code>
	275, 286, 297,	<code>\pUseExpFont@nss</code> 490
	312, 320, 331,	 1170, 1248	<code>\repeat</code>
	342, 353, 1263 , 1274	<code>\pUseKeyFont@nss</code>	. 1248 1133,
<code>\pFalse@nss</code>	<code>\put</code> 1339–1342,	1586, 1616, 1878
	.. 1261 , 1264, 1268		1400, 1406,	<code>\RequirePackage</code>
<code>\pFonts</code>	156, 159, 167,		1469, 1472,	... 441, 445, 447–
	170, 178, 181,		1486, 1487,	449, 1049–1051,
	212, 215, 223,		1490, 1491,	1102, 1104, 1106
	226, 234, 237,		1503, 1504,	<code>\return</code>
	268, 271, 279,		1506, 1507, 1495
	282, 290, 293,		1544, 1554,	<code>\rightskip</code>
	324, 327, 335,		1597, 1600, 1297
	338, 346, 349, 1153		1619, 1639,	<code>\rmdefault</code>
<code>\pKey</code> 142,		1641, 1653, 1114
	161, 172, 183,		1663, 1698,	S
	189, 198, 217,		1699, 1713,	<code>\savlength@nss</code>
	228, 239, 245,		1726, 1727,	1277 ,
	254, 273, 284,		1759, 1779,	1319, 1324,
	295, 301, 310,		1797, 1807,	1334, 1356, 1365
	329, 340, 351,		1817, 1833,	<code>\sBoolValue</code>
	357, 422–430, 1173		1853, 1868, 1270
<code>\pKeyFont@nss</code>	. 1150 ,		1871, 1931,	<code>\section</code>
	1160, 1235, 1253	<code>\pVar</code>	. 141, 143, 160,	138, 194, 250, 306
<code>\pKeyword</code>	. 142, 161,		162, 171, 173,	<code>\selectfont</code>
	172, 183, 198,		182, 184, 189,	. 1129, 1232
	217, 228, 239,		197, 199, 216,	<code>\selectlanguage</code>
	254, 273, 284,		218, 227, 229,	... 36
	295, 310, 329,		238, 240, 245,	<code>\selectlanguageEnglish</code>
	340, 351, 1173 , 1272		253, 255, 272,	. 787, 788, 792,
				793, 819, 820, 823
				<code>\set@nss</code>
			 1381 ,
				1461, 1478,
				1500, 1679,
				1705, 1716, 1911
				<code>\setboolean</code>
			
				... 1058–1061,
				1065–1068,
				1072–1075,
				1080, 1083, 1086
				<code>\sf</code>
			 500, 503
				<code>\sFalse</code>
				... 144, 152,
				163, 174, 185,
				200, 208, 219,
				230, 241, 256,

264, 275, 286,	<code>\sVar</code>	141,	1744,	1763,
297, 312, 320,		160, 171, 182,	1831,	1869,
331, 342, 353, <u>1271</u>		197, 216, 227,	1871, 1873, 1874	
<code>\sffamily</code>		238, 253, 272,	<code>\tempxxx@nss</code> . . <u>1280</u> ,	
. 156, 159, 212,		283, 294, 309,	1542–1544,	
215, 268, 271,		328, 339, 350, <u>1271</u>	1552–1554,	
324, 327, 1150–1152	<code>\switch</code> . . .	43, 45, <u>1729</u>	1752–1754,	
<code>\shape</code> . . 1571, 1577,	<code>\symitalics</code> .	1136, 1137	1759, 1772–	
1584, 1604,			1774, 1779,	
1609, 1614, 1626			1789–1791, 1797	
<code>\sKey</code> 142,				
161, 172, 183,	<code>\T@pageref</code>	451	<code>\tempy@nss</code> <u>1280</u> , 1391,	
198, 217, 228,	<code>\T@ref</code>	450	1394, 1395,	
239, 254, 273,	<code>\tabcolsep</code>	1419	1461, 1462,	
284, 295, 310,	<code>\tempx@nss</code> <u>1280</u> , 1439,		1479, 1480,	
329, 340, 351, <u>1271</u>	1460, 1461,		1498–1500,	
<code>\slshape</code> . . 167, 170,	1467, 1468,		1534, 1536,	
223, 226, 279,	1471, 1477,		1538, 1548,	
282, 335, 338, 1152	1478, 1505–		1550, 1557,	
<code>\small</code> . . 521, 1150–1152	1507, 1524,		1558, 1585,	
<code>\smallskipamount</code> . .	1525, 1530,		1602, 1603,	
. 1327, 1337	1533, 1542,		1615, 1626,	
<code>\spaceskip</code> 1298	1544, 1594–		1650–1652,	
<code>\sProofOff</code> <u>1345</u>	1597, 1634,		1661, 1662,	
<code>\sProofOn</code> 87, <u>1345</u>	1638–1640,		1665, 1667,	
<code>\str@kt</code> . 1329, 1343,	1642, 1655,		1680, 1681,	
1359, <u>1369</u> ,	1677–1679,		1686, 1691,	
1642, 1646,	1704, 1705,		1692, 1696,	
1687, 1710,	1708, 1710,		1699, 1706,	
1720, 1838, 1855	1715, 1716,		1707, 1710,	
<code>\sTrue</code> . 96, 144, 152,	1762–1764,		1711, 1713,	
163, 174, 185,	1812, 1813,		1717, 1719,	
200, 208, 219,	1815, 1817,		1721, 1722,	
230, 241, 256,	1818, 1822,		1727, 1746,	
264, 275, 286,	1823, 1835,		1748, 1750,	
297, 312, 320,	1838, 1851,		1755, 1765,	
331, 342, 353, <u>1271</u>	1854, 1855,		1767, 1769,	
<code>\StruktBoxHeight</code> 86,	1873, 1910,		1775, 1785,	
96–98, 100–104,	1911, 1924,		1787, 1792,	
106–110, 112–	1926, 1935, 1942		1825, 1827,	
116, 119–123, 125	<code>\tempxx@nss</code>		1828, 1830,	
<code>\StrukTeX</code> <u>1108</u>	. . . <u>1280</u> , 1384–		1832, 1835,	
<code>\struktex@out</code>	1386, 1439,		1852, 1854,	
. 508, 512, 515,	1440, 1468–		1911, 1912,	
517, 551, 555, 559	1472, 1526,		1916, 1919,	
<code>\struktoqramm</code> <u>1307</u>	1527, 1531,		1920, 1927,	
<code>\struktoqramm@NoProof</code>	1552, 1554,		1929, 1935, 1942	
. . . <u>1321</u> , 1351, 1353	1598, 1600,		<code>\tempyy@nss</code> . . . <u>1280</u> ,	
<code>\struktoqramm@nss</code> .	1635–1637,		1523, 1529–	
. . . 1311, 1312,	1641, 1646,		1531, 1534,	
1347, 1351, 1353	1665, 1684,		1540, 1556,	
<code>\struktoqramm@Proof</code>	1685, 1687,		1557, 1565,	
. . . 1311, <u>1331</u> , 1347	1709, 1710,		1566, 1595,	
<code>\sub</code> <u>1464</u>	1718–1720,		1596, 1598,	
	1732, 1738,		1601, 1602,	

v2.0	General: Dorothea Rieger, Rechenzentrum RWTH Aachen 1	v3.1a	General: Verschiedene Erweiterungen 1
	Erweitert um ‘exit, ‘forever und ‘foreverend, D. Rieger 3		<code>\CenterNssFile</code> : aus <code>JHfMakro.STY</code> übernommen 66
	<code>\exit</code> : Eingeführt durch D. Rieger 55		<code>\endcenterNss</code> : aus <code>JHfMakro.STY</code> übernommen 66
	<code>\forever</code> : Eingeführt durch D. Rieger 60		<code>\struktogramm</code> : optionaler Parameter eingeführt 50
	<code>\foreverend</code> : Eingeführt durch D. Rieger 60		<code>\struktogramm@NoProof</code> : als internes Makro zur Behandlung optionaler Parameter neu 51
v2.0c	<code>\pKey</code> : neu definiert 46	v3.1b	<code>\CenterNssFile</code> : erheblich vereinfacht 66
	<code>\pKeyword</code> : ‘pKeyword und ‘pKey definiert 46		<code>\endcenterNss</code> : erheblich vereinfacht 66
	<code>\pVar</code> : ‘pVar als Abkürzung definiert 46	v3.2a	General: Beschreibung eingeführt 53
v2.0n	General: Dokumentation überarbeitet 6		Dokumentation verbessert 1
	<code>\pKeyword</code> : Neuimplementation gemäß ‘TeXnischer Komödie’ 46		<code>\assign</code> : Beschreibung eingeführt 55
	<code>\pVariable</code> : Neuimplementation gemäß ‘TeXnischer Komödie’ 46		<code>\gsize@nss</code> : Beschreibung eingeführt 49
v2.1	General: ‘exit modifiziert, J. Hoffmann 3		<code>\gx@nss</code> : Beschreibung eingeführt 49
v2.2	General: Beschreibung von Variablen eingeführt, J. Hoffmann 3		<code>\param@nss</code> : Beschreibung eingeführt 49
v2.2a	General: Beschreibung von Variablen verbessert, J. Hoffmann 3		<code>\TextBoxHt@nss</code> : Beschreibung eingeführt 50
v2.5	General: Jobst Hoffmann, Rechenzentrum RWTH Aachen 1		<code>\ydepth@nss</code> : Beschreibung fehlt noch 49
	<code>\exit</code> : Änderungen an der Darstellung 55	v3.3a	General: Schnittstellenbeschreibung verallgemeinert 1
	<code>\filler@nss</code> : Leerraum von ‘ auf ‘emptyset umgestellt 50		<code>\descriptionindent</code> : neu eingeführt 54
	<code>\savelength@nss</code> : ‘VarDeclWd eingeführt 49		<code>\descriptionsep</code> : neu eingeführt 54
	<code>\sVar</code> : eingeführt durch Jhf 49		<code>\descriptionwidth</code> : neu eingeführt 54
v3.0a	General: Urversion mit integrierter Dokumentation 1		<code>\struktogramm@NoProof</code> : Opt innerhalb von Struktogrammen korrigiert 51
v3.1	<code>\anzzeilen@nss</code> : Beschreibung eingeführt 56	v3.3c	<code>\exit</code> : Fehler mit der Breite korrigiert 55
	<code>\c@nter@nss</code> : Beschreibung eingeführt 50		<code>\pFalse</code> : Beschreibung eingeführt 48
	<code>\ifthenelse@nss</code> : Beschreibung eingeführt 57		<code>\pTrue</code> : Beschreibung eingeführt 48
			<code>\x@nss</code> : Beschreibung eingeführt 49
			<code>\xsize@nss</code> : Beschreibung eingeführt 49
			<code>\y@nss</code> : Beschreibung eingeführt 49
		v3.3e	<code>\CenterNssFile</code> : ‘TmpBox umbenannt in ‘nss@CenterBox 66

‘centerline durch ‘strut‘hfill ... er-		\y@nss: umbenannt von ‘y	49
setzt	66	\ydepth@nss: umbenannt von	
Auf zusätzliche Gruppe um ‘triv-		‘ydepth	49
list verzichtet	66	\yy@nss: umbenannt von ‘yy . . .	49
\endcenterNss: ‘TmpBox umbe-		v4.0c	
nannt in ‘nss@CenterBox	66	\struktogramm@NoProof: umbe-	
‘centerline durch ‘strut‘hfill ... er-		nannt von ‘nss@struktogramm	51
setzt	66	v4.0d	
v3.5a		\ifthenelse@nss: Tippfehler korri-	
General: Entwicklungslinien J. Die-		giert	57
tel, J. Hoffmann vereinigt	1	v4.1a	
\forever: übernommen von J. Die-		General: Geraden mit beliebiger	
tel	60	Steigung durch ‘emline möglich	1
\foreverend: übernommen von J.		\btt: ‘btt auf NFSS umgestellt .	44
Dietel	60	\complex: ‘nat, ‘integer und ‘real	
\switch: übernommen von J. Die-		eingeführt	44
tel	61	\filler@nss: ‘emptyset wird als	
v3.5b		AMS-Schrift geladen	50
\pFalse: umbenannt von ‘false zur		umbenannt von ‘filler	50
Vermeidung von Interferenzen	48	\MathNormal: ‘MathItalics und	
\pTrue: umbenannt von ‘true zur		‘MathNormal eingeführt	44
Vermeidung von Interferenzen	48	\sKey: ‘btt bereits anderweitig defi-	
\sKey: neu eingeführt	49	niert	49
\sVar: umbenannt von ‘Var zur		\TextBox@nss: umbenannt von	
Vermeidung von Interferenzen	49	‘nss@Box	50
v4.0a		\TextBoxHt@nss: umbenannt von	
General: Anpassung an LaTeX2e .	1	‘nss@BoxHt	50
v4.0b		v4.1b	
General: umbenannt von ‘xin . . .	53	General: Höhe des Bedingungs-	
\anzeilen@nss: umbenannt von		rechteckes bei ‘ifthenelse vor-	
‘zeilen	56	gegebenbar	1
\c@nter@nss: umbenannt von		\anzeilen@nss: umbenannt von	
‘c@nter	50	‘zeilen@nss	56
\condindent@nss: umbenannt von		\complex: ‘complex eingeführt . .	44
‘w@rt	56	\condindent@nss: umbenannt von	
\gsize@nss: umbenannt von ‘gsize	49	‘w@rt@nss	56
\gx@nss: umbenannt von ‘gx . . .	49	\index@nss: umbenannt von ‘zaeh-	
\gy@nss: umbenannt von ‘gy . . .	49	ler@nss	56
\index@nss: umbenannt von ‘zaeh-		v4.1c	
ler	56	General: Dokumentation verbes-	
\param@nss: umbenannt von ‘pa-		sert	1
ram	49	Eigene Datei für spezielle Fonts	
\savelength@nss: umbenannt von		anlegen	44
‘s@velength	49	Eigene Datei zum Austesten der	
\set@nss: umbenannt von ‘set . .	53	Makros integriert	21, 22
\temp@nss: umbenannt von		v4.1d	
‘temp	49	\ifthenelse@nss: Tippfehler korri-	
\tempxx@nss: umbenannt von		giert	57
‘tempxx	49	\switch: Dokumentation verbes-	
\tempy@nss: umbenannt von ‘tem-		sert	61
py	49	v4.2a	
\@x@nss: umbenannt von ‘x	49	\cases@nss: neu eingeführt	61
\@xsize@nss: umbenannt von ‘xsize	49	\switch: um <i>default</i> -Zweig erwei-	
\@xx@nss: umbenannt von ‘xx . . .	49	tert	61

v4.2b		<code>\pCommentFont@nss</code> : neu eingeführt	45
	<code>\sProofOff</code> : neu eingeführt	51	
	<code>\struktogramm@NoProof</code> : umbenannt von <code>'struktogramm@nss</code>	51	
	<code>\struktogramm@Proof</code> : neu eingeführt	51	
v4.3a	General: Dokumentation verbessert und korrigiert	5, 52	
	Variable <code>'sBoolValue</code> ersetzt <code>'sL-Val</code>	2	
v4.4a	General: Kommando <code>'ifthenelse</code> internalisiert	3	
	<code>\ifthenelse@nss</code> : Kommando intern gemacht, um benannt von <code>'ifthenelse</code>	57	
	<code>\struktogramm</code> : <code>'ifthenelse</code> als internes Kommando definiert	50	
v4.4b	General: <code>'RequirePackage</code> umgestellt wg. korrekter Versionsmeldung	42	
	<code>\struktogramm</code> : Optionaler Parameter über <code>'empty</code> statt <code>'relax</code> getestet	50	
	<code>\struktogramm@NoProof</code> : Optionaler Parameter über <code>'empty</code> statt <code>'relax</code> getestet	51	
	<code>\struktogramm@Proof</code> : Optionaler Parameter über <code>'empty</code> statt <code>'relax</code> getestet	51	
v4.5a	General: Eigenes Paket zur Formatierung von Programminhalten integriert	1	
	Umbenennungen: <code>'sVar</code> → <code>'pVar</code> , ...	2	
	Umgebung <code>description</code> auf <code>list</code> -Umgebung zurückgeführt	1	
	<code>\description@nss</code> : neu eingeführt	55	
	<code>\descriptionindent</code> : an <code>L^AT_EX2e</code> angepasst	54	
	<code>\descriptionlabel@nss</code> : neu eingeführt	54	
	<code>\descriptionsep</code> : an <code>L^AT_EX2e</code> angepasst	54	
	<code>\descriptionwidth</code> : an <code>L^AT_EX2e</code> angepasst	54	
	<code>\descriptionwidth@nss</code> : neu eingeführt	54	
	<code>declaration</code> : neu eingeführt	54	
	<code>\pBoolValue</code> : neu definiert	49	
	<code>\pTrue@nss</code> : neu definiert über	48	
	<code>\pFalse@nss</code> : neu definiert	48	
	<code>\pFonts</code> : neu eingeführt	45	
	<code>\pKeyFont@nss</code> : neu eingeführt	45	
	<code>\pTrue</code> : neu definiert über	48	
	<code>\pTrue@nss</code> : neu definiert	48	
	<code>\return</code> : Beschreibung eingeführt	56	
	<code>\sub</code> : Beschreibung eingeführt	55	
v4.5b	<code>\change</code> : Beschreibung eingeführt	60	
	<code>\dfr</code> : Beschreibung eingeführt	61	
	<code>\dfrend</code> : Beschreibung eingeführt	61	
	<code>\ifend@nss</code> : Beschreibung eingeführt	60	
	<code>\struktogramm@NoProof</code> : Positionierung des Struktogramms über [t] bei der minipage	51	
	<code>\struktogramm@Proof</code> : Positionierung des Struktogramms über [t] bei der minipage	51	
v4.5c	<code>\PositionNSS</code> : neu eingeführt zur Positionierung der Struktogramme	50	
v4.5d	<code>\CenterNssFile</code> : Belegen von <code>'CenterBox@nss</code> in das <code>'item</code> verlegt	66	
	<code>\endcenterNss</code> : <code>'CenterBox@nss</code> global gemacht	66	
v4.5e	General: Neue <code>.sty</code> -Datei <code>strukt.doc.sty</code> eingeführt	3	
	Verzicht auf Groß- und Kleinschreibung im Namen von <code>.sty</code> -Dateien	1	
v5.0	General: Abhängigkeiten der Dokumentation von zusätzlichen Paketen beseitigt	1	
	Dateinamen zwecks Kompatibilität auf Kleinschreibung umgesetzt	1	
	kleinere textuelle Änderungen, Fontauswahl teilweise geändert	1	
v5.1a	General: kleinere stilistische Änderungen nach W. Hanrath	1	
v5.1b	General: Umstellung der Dokumentation mittels Umgebung		

	example, ‘verbatiminput und verbatimwrite	1		Umgebungen zur einfacheren Notation der Dokumentation eingeführt	1
v5.2a	General: kleinere Korrekturen	1	v6.0a	General: Code umgestaltet, z.B. ‘def → ‘newcommand, \$...\$ → ‘(...‘)	1
	Umstellung der Dokumentation, dadurch Verzicht auf eigene Treiberdatei möglich	1		Dokumentation von ‘case verbessert	1
	\descriptionlabel@nss: Vorbesetzung des Fonts entfällt	54		Englische Übersetzung der Dokumentation hinzugefügt	1
v5.2b	General: Tippfehlerkorrektur: Umlaut „ue“ durch „ü“ ersetzt	1		Makros ‘cs, ‘marg, ‘oarg, ‘pparg sowie ‘envb und ‘enve zur besseren Dokumentation eingesetzt	1
v5.3a	General: Die Form des Textes in Bedingungen ist nun an beliebige Steigungen angepasst	1		Zusätzliche Paketoption „curves“ zur Gestaltung beliebiger Steigungen eingeführt	1
	Option emlines eingeführt	42	v7.0	General: Wechsel von der RWTH Aachen zur FH Aachen, Abt. Jülich	1
	\indentmeasure@nss: neu eingeführt	56	v7.0a	General: ‘pLanguage geändert: ‘pLanguage@nss zusätzlich eingeführt	1
	\indentmeasureleft@nss: neu eingeführt	56		Beispiele für unterschiedlichen Programmierspracheneinsatz erweitert	1
	\thisindent@nss: neu eingeführt	56		Code teilweise umgestellt	1
	\thisindentleft@nss: neu eingeführt	56		Dateinamen geändert	1
	\thislength@nss: neu eingeführt	56		Makefile integriert	1
v5.4a	General: stukt xp.sty überarbeitet zur besseren Darstellung von Programmen	1		Marker für interne Beispiele umbenannt	1
	\Cee@nss: neu definiert	47		Option verification eingeführt	43
	\CheckForKeyword@nss: neu definiert	47		Sprachbezeichnungen geändert: Suffix @nss	1
	\Java@nss: neu definiert	47		Testrahmen geändert	1
	\LaTeX@nss: neu definiert	47		Tippfehler korrigiert	1
	\Pascal@nss: neu definiert	47		\assert: neu eingeführt	66
	\pExp: ‘pExp als Abkürzung definiert	46		\assert@nss: neu eingeführt	66
	\pExpFont@nss: neu eingeführt, ersetzt ‘pVarFont@nss	45		\assertblock@nss: neu eingeführt	67
	\pExpression: neu eingeführt, ersetzt ‘pVariable	46		\block@nss: umbenannt von ‘bl@ck	53
	\pKey: auf ‘pVariable zurückgeführt	46		\xsizeo@nss: neu eingeführt	66
	\pKeyword: auf ‘pVariable zurückgeführt	46		\ydeptho@nss: neu eingeführt	66
	\pLanguage: neu definiert	47	v7.0b	General: Anzahl der Zähler reduziert	1
	\pUseExpFont@nss: neu eingeführt, ersetzt ‘pUseVarFont@nss	48		ersetzt durch ‘tempxx@nss	53
	\pVariabl@: neu definiert	46		\cases@nss: ersetzt durch ‘tempxx@nss	61
	\pVariable: neu definiert	46	v7.0ba	\gindhelp@nss: umbenannt von ‘indhelp@nss	56
v5.4b	General: Erster Versuch einer gemischtsprachigen Dokumentation	1			

v7.2		v8.0e	
	<code>\tempxxx@nss</code> : eingeführt für die Unterstützung von <code>pict2e</code> . . . 49	General: & mit Fluchtzeichen versehen 34	
	<code>\tempyy@nss</code> : umbenannt von ‘tempyy’ 49	nun wird die .pdf-Ausgabe sauber abgefangen 29	
	<code>\tempyyy@nss</code> : eingeführt für die Unterstützung von <code>pict2e</code> . . . 49	Tippfehler korrigiert: <code>\$\$\$f</code> → <code>\$\$f</code> 34	
v8.0a		<code>\ifthenelse@nss</code> : Fehlermeldung erweitert, nun auch Hinweis auf <code>curves</code> und <code>pict2e</code> 58	
	<code>\ifthenelse@nss</code> : Paket <code>pict2e.sty</code> wird unterstützt 57	<code>\switch</code> : ‘tempxxx@nss’ umbenannt in ‘tempxx@nss’ (wir befinden uns nicht im Fall <code>pictIIe</code>) 64	
	<code>\switch</code> : Paket <code>pict2e.sty</code> wird unterstützt 61		
v8.0b		v8.0f	
	General: .pdf-Ausgabe eingeführt ‘fileversion’ und ‘filedate’ zur einfacheren zentralen Verwaltung von Entwicklungsdaten eingeführt 1	General: Dokumentation von ‘case’ verbessert 1	
	Entwicklungsgeschichte korrigiert und erweitert 2	<code>\switch</code> : ‘tempxx@nss’ umbenannt in ‘tempx@nss’ (die Anzahl der Fälle wurde versehentlich geändert) 64	
	Hinweis auf <code>pict2e.sty</code> 2		
	Umstellung der Sprachverwaltung auf <code>babel</code> 4	v8.1a	
	Zeichnung zum Layout der Deklarationen verbessert (‘multiput’ statt n-mal ‘put’, Korrektur von Bezeichnungen) 10	General: Erste (fehlerhafte) Version einer <code>struktex.el</code> Stil Datei zur Unterstützung von <code>AUCTEX</code> hinzugefügt 1	
	<code>\btt</code> : Hinweis auf Luxi Mono als Ersatz fuer <code>cmbtt</code> 44	v8.2a	
v8.0c		General: <code>HOMETEXMF</code> → <code>TEXMFHOME</code> 34	
	General: Zweisprachigkeit verbessert, Bezeichner statt Nummern eingeführt 1	Option <code>nofiller</code> eingeführt . . 43	
v8.0d		<code>struktex.el</code> korrigiert und erweitert 1	
	General: ‘switch’ zeigt bei geeigneten Beispielen das rechtsbündige Setzen von Bedingungen . . . 1	<code>\filler@nss</code> : ‘ifmmode’ → ‘ensuremath’ 50	
	<code>\pVariabl@</code> : Python-Schlüsselwörter neu eingeführt 46	<code>\switch</code> : Fehlerkorrektur: schräge Linie endet nun an der richtigen Stelle (<code>pict2e</code>) 61	
	<code>\Python@nss</code> : Python als zulässige Sprache neu definiert 47		
	<code>\switch</code> : Fehlerkorrektur: senkrechte Trennlinien haben nun die richtige Länge 61	v8.3	
		General: Optionen <code>draft</code> und <code>final</code> eingeführt 43	
		<code>struktex.el</code> erweitert, so dass die Parameter alle Kommandos automatisch abgefragt werden. . 38	
		Weitere Optionen „draft“ und „final“ eingeführt 1	