BLUe's Typesetting of Pascal

Lean and Mean

Kees van der Laan

Hunzeweg 57,
9893 PB Garnwerd, The Netherlands
cgl@rc.service.rug.nl

Abstract

The formatting of (partial) PASCAL programs within plain TEX is proposed. Only `\beginpascal` and `\endpascal` have to be added as markup. In general the literate programming tools are to be preferred, especially when designing, developing, documenting, and maintaining professional software.

Keywords: Compatible extension, education, fifo, macro writing, Pascal texts, pattern matching, plain TEX, pretty-printing, reusable software parts, software engineering.

1 Introduction

Programs are meant for computers and humans to be understood. Therefore publishing a program is not just a verbatim listing. In general the typesetting of a program is part of the documentation process.

Knuth with his (CWEB) literate programming tools aimed at programming as a craft, to program

- with less errors, and
- to create documentation simultaneously.

Pretty-printing of code as such is an academic exercise. Spending too much energy on it is suboptimization, because of the wealth of already available tools.

1.1 Why?

Knuth, *The TEXbook* 234 and ex18.9, illustrates the problem via a small program fragment, especially in view of non-monospaced types. He adds tabular markup to secure vertical alignment, and inserts tags for marking up for fonts. If only program fragments are the subject, this way of doing is OK. The larger the code the more tedious hand markup becomes. Moreover, detailed hand markup is inconsistency-prone.

I strove after to leave the program as such unaltered, and assumed that the Alignment has already been taken care of. The fonts for the reserved words, the program text, and can be adaptedly easily by the user.

1.2 Disclaimer

Because I refrained from inserting markup in the program, vertical alignment can be hampered, due to the use of the non-monospaced fonts and different representations of some symbols like $\geq$.

1.3 Notations and definitions

`\ea` denotes `\expandafter`. `\nx` denotes `\noexpand`. FIFO stands for First In First Out, as described in my 'FIFO and LIFO sing the BLUes.'

2 Use

Within the context of blue.tex the following example[1]

{ Print length of third side of triangle

given two sides and enclosed angle }

program *EX4A(input, output)*;

var *a, b, c, angle* : *real*;

begin *read(a, b, angle)*;

   c := *sqrt(sqr(a)* + *sqr(b)* − 2 ∗ *a* ∗ *b* ∗ *cos(angle))*;

   writeln('The third side is', c)

end.

results from

```
\beginpascal
{Print length of third side of triangle
 given two sides and enclosed angle}
program EX4A(input, output);
var a, b, c angle : real;
begin read(a, b, angle);
   c:=sqrt(sqr(a) + sqr(b) - 2*a*b*cos(angle));
   writeln('The third side is', c)
end.
\endpascal
```

With the token variables `\prepascal` and `\postpascal` the placement within context can be altered. For example to 'center' the program use the following.

```
\prepascal{$$\vbox\bgroup\hsize.5\hsize}
```

[1] Courtesy Wilson & Addyman.

```
\postpascal{\egroup$$}
\beginpascal
⟨code⟩
\endpascal
```

3 What are the problems?

Given that the layout in ASCII is already there, then the vertical alignment is violated with non-monospaced fonts.[2]

If we adopt Knuth's approach to add markup for alignment then the program is altered, and it doesn't compile as such.[3]

Doumont proposed to use TEX as a pre-processor, a full-blown pretty-printer. Tradition has it that the layout is done by other tools than TEX. Doumont's approach is therefore not simple, and not easily adaptable to other languages. My modest approach is hopefully a good balance between Knuth's markup of program fragments and Doumont's pretty-printer.

4 Coding

My approach is to process a program line-by-line, and each line word-by-word, an application of nested FIFO processing.

Comments and special symbols can do their job by making them active and defining them appropriately.

5 Customization

The user can choose his fonts. Default is provided

```
\let\reservedwordsfont\bf
\let\pascalprogramfont\it
\let\pascalcommentfont\rm
```

The setup can be used for similar languages by adapting `\reservedwords` and the macros for typesetting the special symbols.

6 Availability

The macros — ≈ 75 lines — are shareware and come with blue.tex, i.e., are included in the tools.dat database.[4]

7 Acknowledgements

Erik Frambach thank you for your assistance, and your sound judgement. As usual Jos Winnink helped me to procrust the markup of the article into `maps.sty`. Jean-Luc Doumont prompted the subtitle.

8 Conclusion

In general it is difficult to know when to stop, as Schumacher in his precious 'Small is beautiful' already pointed out. Perhaps, I stopped too early, but the macros do what I want them to do. My case rest.

9 References

I learned from Doumont, but my approach is much different. Look in lit.dat for more details of consulted work.

10 Appendix: The macros

To facilitate the use blue.tex contains the storage allocations and the following.

```
\def\beginpascal{\begingroup\let\pascaltool=x
   \loadtool\afterassignment\fifol\let\dummy=}
```

As part of tools.dat are provided the blue collar macros. Note that I included in the set of reserved words `\end.` and `\end;` .

In the following driver I have collected the items from blue.tex and tools.dat to provide a complete and independent set to be used with AnyTEX.

```
%Declarations
\newtoks\prepascal
\newtoks\postpascal
\newtoks\reservedset
\newcount\commentstatus
\newif\iffound
%Initializations
\let\reservedwordsfont\bf
\let\pascalprogramfont\it
\let\pascalcommentfont\rm
\prepascal{\medskip\noindent}
\postpascal\prepascal
\let\ea\expandafter
%Auxiliary
\newif\iffound
\def\loc#1#2{\def\locate##1#1##2\end
 {\ifx\empty##2\empty\foundfalse
 \else\foundtrue\fi}\ea\locate#2.#1\end}
%
\def\beginpascal{\begingroup\parindent0pt
%   \let\pascaltool=x \loadtool
   \afterassignment\fifol\let\dummy=}
{\obeylines\gdef\fifol#1
 {\ifx\lofif#1\lofif\fi%
  \processl{#1}\fifol}}
\def\wofif#1\fifow{\fi}
%From tools.dat
\def\lofif#1\fifol{\fi
   \the\postpascal\endgroup}
\let\endpascal\lofif
\def\processw#1{%
 \ifnum\commentstatus>0 #1\else
  \ifx\relax#1\relax{ }%space
  \else\foundfalse
   \loc{#1}{\the\reservedset}%
    \iffound{\reservedwordsfont#1}%
    \else\if=#1${}={}$\else
     {\pascalprogramfont#1{}}\fi\fi
 \fi\fi}%end processw
%Typesetting : ;= ; . ˆ
\def\becomes{:=}
\catcode‘\:=13
\def:#1{\if=#1${}\becomes{}$\else
             {\rm\char‘\:}#1\fi}
\catcode‘\;=13
```

[2] I don't care because readability is not hampered. The rule for indentation will not be obeyed.

[3] I know of Rein Smedinga's argument that for small educational fragments this is not a problem.

[4] blue.tex is on NTG's 1995 4AllTEX CD-ROM, and will be offered to the CTAN. Each happy user from blue.tex is requested to send me $25 to maintain my hardware. The donor will be added to my register.blu database.

```
     \def;{{\/\rm\char‘\;}}
\catcode‘\.=13 \def.{{\pascalprogramfont
                     \char‘\.}}
\catcode‘\^=13 \def^{\char’136}
    %\def^{{\enspace$\uparrow$}}
%Reserved words
\reservedset{and array begin case const
 div do downto else end. end; file for
 function goto if in label mod nil not of
 or packed procedure program record repeat
 set then to type until var while with}%
%Handling of >= <= <> > <
\def\lt{<}\def\gt{>}
\catcode‘\>=13
\def>#1{\if=#1${}\geq{}$\else
             ${}\gt{}$#1\fi}
\catcode‘\<=13
\def<#1{\ifx>#1${}\ne{}$\else
       \if=#1${}\leq{}$\else
       ${}\lt{}$#1\fi\fi}
%Handling of + - *
\def\-{-}%Save for hyphen use
\def\plus{+}\def\minus{-}\def\times{*}
\catcode‘\+=13 \def+{${}\plus{}$}
\catcode‘\*=13 \def*{${}\times{}$}
%{\catcode‘\-=13 \gdef-{${}\minus{}$}}
%Later catcode is set for minus
%Last part of processing
\obeyspaces\let =\
\def\fifow#1 {\ifx
\wofif#1\wofif\fi\processw{#1}\ \fifow}%
\def\processl#1{\fifow#1 \wofif
 \endgraf}%First space is needed
%
%Keeping track of comments, and
%using comment fonts
\catcode‘\{=13\catcode‘\}=13%
\catcode‘\[=1\catcode‘\]=2%
\gdef{[$\{\,$\global%
\advance\commentstatus1\relax%
\pascalcommentfont]%
\gdef}[$\,\}$\global%
\advance\commentstatus-1]%
\catcode‘\-=13 \def-[$[]\minus[]$]%
\obeylines\smallskip%
\the\prepascal\relax%
```