

Literate Programming

in *The Encyclopedia of Computer Science*, A. Ralston and E. D. Reilly and D. Hemmendinger, eds., 4th edition, 1000–1002, Nature Publishing Group, 2000.

For articles on related subjects *see* DOCUMENTATION, MACRO-INSTRUCTION, PRETTY-PRINTING, STRUCTURED PROGRAMMING, T_EX.

A programmer finds documentation difficult enough without trying to make it correspond to the program! *Literate programming* is a system of combining program and internal documentation, so that both may be developed closely together with ease. Various automatic aids to readability are provided, such as substantial cross-referencing and indexing, pretty printing and so on. Special macro processing allows the program to be written in any order to improve and simplify the exposition; macros are automatically numbered so that their usage is easily cross-referenced.

An important point is that a literate programming system converts the documentation and code into beautifully typeset material *with no additional effort* by the programmer. All features combine to simplify and encourage the documentation process and to keep it in very close correspondence with the actual program. Efficient means are provided to extract the program code from the literate program so that it may be compiled or processed in the usual ways.

	⋮
	@* Insert sort
Extract from the source	This is the standard insert sort algorithm.
of a literate program —	Assumes a sentinel at \$a[0]\$.
illustrating interleaving	@p
of code, documentation	for i := 2 to N do
and macros.	begin v := a[i]; j := i;
	@<Insert...>
	end
The @ signs are this	@
system's convention for	@<Insert \$v\$ in the array@>=
introducing various	while a[j-1]>v do
literate programming	begin a[j] := a[j-1]; j := j-1 end;
features.	a[j] := v
	⋮

The example shows the use and definition of a macro called `<Insert v in the array>`. Macro names may be abbreviated (e.g., `<Insert...>`), helping encourage programmers to use longer, more mnemonic, names. The \$ symbols tell the literate programming system to typeset and cross-reference certain text as code rather than as commentary. The result of processing this example fragment creates the source code and typeset versions shown in the Figure.

```

:
:
for i:=2 to N do
begin v:=a[i];j:=i;
while a[j-1]>v do
begin a[j]:=a[j-1];j:=j-1 end;
a[j]:=v
end
:
:

```

Generated code.

```

:
:
31. Insert sort. This is the standard insert
sort algorithm. Assumes a sentinel at a[0].
for i := 2 to N do
begin
  v := a[i]; j := i;
  ⟨Insert v in the array. 32⟩
end
32. ⟨Insert v in the array. 32⟩≡
while a[j - 1] > v do
begin a[j] := a[j - 1]; j := j - 1 end;
a[j] := v
Used in section 31.
:
:

```

Typeset program.

*Not showing the automatically
generated table of contents, index etc.*

There is no need for literate programming to be confined to batch programming languages and conventional documentation. The idea of combining different types of text to make them easier to maintain together is quite general: reference [3] exhibits a runnable Pascal program documented in Japanese; while an obvious application is the combination of formal specification with conventional program. The symbolic mathematics system *Mathematica* [4] uses a form of literate programming in its ‘notebooks’: it allows mathematical articles to be written mixing text with mathematics which can be evaluated. Other possibilities and advantages are suggested in [2].

Literate programming was developed by D. E. Knuth in the late 1970s [1] and has been used most successfully in the implementation and documentation of his large typesetting system \TeX . The clarity resulting from the technique has been instrumental in the range of its successful implementations. Knuth’s programs are unique in being published in their entirety as readable books—of course, other programs have been published, but not readably and not in *exactly* the form in which they may be compiled and run.

References

- [1] D. E. Knuth, *Literate Programming*, CSLI Lecture Notes Number 27, Centre for Study of Language and Information, 1992.
- [2] H. W. Thimbleby, Experiences of ‘Literate Programming’ using `cweb` (a variant of Knuth’s `WEB`), *Computer Journal*, **29**, 3, pp201–211, 1986.
- [3] T. Kurokawa, Literate Programming, *bit*, **17**, 4, pp426–450, 1985 (pub. Kyoritsu Shuppan). Japanese translation of Knuth’s original paper (reprinted in [1]).
- [4] S. Wolfram, *Mathematica*, Addison-Wesley, 1988.

Index of terms

documentation; literate programming; macro processing; *Mathematica*; pretty printing; T_EX.

HAROLD W. THIMBLEBY