Literate Programming in WinWord*

Lee Wittenberg
Tipton Cole+Company
3006 Bee Cave Road, Suite D–200
Austin, TX 78746
(512) 329-0060
leew@pilot.njin.net

Disclaimer

This program is provided merely as a jumping-off point for future work by others, and therefore comes with no warranty whatsoever. *The author has no intention of supporting it, as he does not use it himself.* This work is freely distributable, and it is expected that works derived from it will be freely distributable as well. If you create a work based on this program that you intend to release commercially, you must first get permission from the author. If your work is to be distributed at no charge, no special permission is necessary, but you must give appropriate credit to Lee Wittenberg and Tipton Cole+Co. in your documentation and copyright (or copyleft) notices..

Introduction

WinWordWEB is a simple collection of WordBasic "macros" that provide a crude literate programming environment. The "look and feel" of the system is based on Norman Ramsey's `noweb`, but can easily be modified to suit individual taste.

Using WinWordWEB

The WinWordWEB system consists of 3 macros (`chunkdefn`, `chunkname`, and `tangle`) and a paragraph style (`code`) contained in the `WORDWEB.DOT` template. Each of these is made available via menu selections and "hot keys" (as described below) so there is no need to access the macros directly.

A WinWordWEB document is a combination of plain text and code chunks. Code chunks are paragraphs formatted with the "`code`" style. These code chunks are "tangled" to create a program that can then be compiled by a traditional compiler. A code chunk looks something like this:

```
⟨A typical Pascal program⟩ ≡
PROGRAM Typical(input, output);
VAR
    ⟨Variables needed by the program⟩
BEGIN
    ⟨Body of the program⟩
END.
```

The "⟨*A typical Pascal program*⟩ ≡" is required, and is inserted into a code chunk using the "Insert|Chunk Def'n" menu item (or by using the Control-Shift-D key combination). Any code chunk that does not begin with a definition line will be ignored when the program is tangled. "Insert|Chunk Def'n" inserts the two angle brackets and the '≡' symbol followed by a soft newline (Shift-Enter). It then puts an asterisk inside the brackets, and formats it as Times New Roman Italic, leaving the asterisk selected so you can type your own chunk name. When you've finished typing the chunk name, the ↓ key will move the cursor to the blank line below the name, where you can begin typing the body of the chunk definition.

---

* A.k.a. Microsoft Word for Windows™

Within the body of a code chunk, you can use the "Insert|Chunk" menu item (or Control-Shift-K) to insert the name of another chunk (e.g., ⟨*Body of the program*⟩, above), which will be expanded to the appropriate definition when the program is tangled. "Insert|Chunk" works similarly to "Insert|Chunk Def'n" in that it inserts the angle brackets and a selected, formatted asterisk over which you can type the desired chunk name.

Tangling

You tangle a program by choosing the "File|Tangle" menu item (or Control-Shift-G). A dialog box will be displayed, allowing you to choose the name of the file in which to put the tangled output. The default name (shown in the box) is the name of the current file with a "TXT" extension replacing the original "DOC." You may then change the chosen filename, if you wish, before tangling starts. You can use the "File|Set Tangle Extension" menu item to change the default extension, if you don't like "TXT," as well.

As tangling begins, the code chunks are all put into a table. Code chunks with the same name are concatenated, in order of appearance. When the table is complete, the tangle macro looks for the code chunk named "⟨*⟩". If there is no such code chunk, an error message is displayed, and nothing will be written to the output file. Otherwise, the text of the ⟨*⟩ chunk will be written to the output file. Each chunk name in the text will be expanded (and written to the output file) as it is encountered, and so on, recursively, for each chunk name referenced in the expanded chunk. If a chunk is referenced that does not have a definition, an error message is displayed, but tangling continues with the null string used as the definition.

Needless to say, all non-code paragraphs are ignored when tangling. Formatting codes within code chunks are likewise ignored, so you may put all your keywords in bold Times Roman, for example, and your comments in Helvetica, if you like.

Implementation Note

Since this is primarily a "Let's see if we can do it" kind of prototype, the algorithms use basically BFI[1] techniques, and error checking is kept to a minimum. The numerous kludges are due both to the author's inexperience with Word and WordBasic, and to bugs discovered therein.

Limitations

- WordBasic is limited to 64K of string data. This effectively limits the size of the tangled output to less than 64K.
- Tangle is rather naive about the angle bracket symbols used to designate the names of code chunks. As a result, no inserted symbols are allowed in code chunks, even within comments. Also, if you begin a code chunk with a chunk name instead of a chunk definition, tangle will get very confused.
- Tangle is also naive about periods in filenames. If a directory somewhere in the current file's path has a period in it, tangle will not generate the proper output filename.
- Code chunk indentation is only occasionally respected in the tangled output, thus making WinWordWEB practically useless for languages like ABC in which indentation is significant.
- Since the Enter key is used in Word to end paragraphs, you have to be very careful always to use Shift-Enter for newlines within a code chunk.
- The macros were designed assuming that Word is in "insert" mode—if you prefer overstrike mode, lotsa luck.

Bugs

I wouldn't be surprised.

---

[1] Brute Force and Ignorance.