```
// SUMMARIZ.WCM -- summarize the contents and location of comments, and
all pieces
//                  of text enclosed between user-defined flags, in the current
document.
//                  Output goes to a loaded doc, which is created in a new
buffer if it is
//                  not already loaded, and which is appended to otherwise.
//                  This means that this macro can be run repeatedly to
summarize several
//                  documents, with all the output accumulating in one place.
//                  Flags and output doc name are defined in the configuration
section below.
//
// AUTHOR:
//      <rreiner@nexus.yorku.ca>
//
// REVISION HISTORY
//      9 Sep 92    RjR    Initial version
//      30 Oct 92   RjR    Handle comments too
//      31 Oct 92   RjR    Parametrization, and documentation improvements

Application (WP;WPWP;Default)


//************************************************************************
// Configuration variables
//
// The flags should be character sequences that do not appear in regular text.
Any length
//      is OK, but two characters or more is a good idea.
// String that signals the beginning of text to include in the summary:
FlagOpen:="[*"
// String that signals the end of text to include in the summary:
FlagClose:="*]"
// Name of the document in which output is accumulated
OutputDocName:="summariz.out"


//************************************************************************
// Main program
//
ONCANCEL(QQuit@)

// We don't want to be prompted for DocSummary info when saving the
output doc, so
//      we save state and then turn summary prompting off.  State is restored
```

```
at exit.
GetWPData(DocSum; DocSummaryPromptOnExit!)
IF (DocSum)
    PrefDocSummary(CreateOnExit:No!)
ENDIF

// Compute useful values for later use
STRLEN(FlagOpLen; FlagOpen)
STRLEN(FlagClLen; FlagClose)

CALL(FindOrCreateTarg@)

// Set up local not-found handler
ONNOTFOUND(CommentsEnd@)
CALL(DoComments@)
// Target for DoComments not-found handler
LABEL(CommentsEnd@)

// Set up local not-found handler
ONNOTFOUND(FlaggedEnd@)
CALL(DoFlaggedText@)
// Target for DoFlagged not-found handler
LABEL(FlaggedEnd@)

GO(QQuit@)

// NOTREACHED


//*****************************************************************************
// FindOrCreateTarg@ -- find the output window, or create it
LABEL(FindOrCreateTarg@)
    // Try to find doc
    GetWPData(InitialCurDoc; CurrentDocument!)
    REPEAT
        DocNext()
        GetWPData(CurDoc; CurrentDocument!)
        GetWPData(CurNam; Name!)
    UNTIL ((CurDoc = InitialCurDoc) OR (CurNam = OutputDocName))

    // Doc was not found; create it.
    IF (CurNam <> OutputDocName)
        FileNew()
        // Check to see if create succeeded
        GetWPData(OutputDoc; CurrentDocument!)
        IF (OutputDoc = CurDoc)
```

```
            // Create failed
            PROMPT("SUMMARIZ.WCM Error"; "Could not create output
document"; 1;;)
            PAUSE
            ENDPROMPT
        ELSE
            // Create succeeded
            FileSave(OutputDocName;WordPerfect51!;Yes!)
            FontLarge(On!)
            Type("Summariz: starting output to " + OutputDocName + " on ")
            DateText()
            FontLarge(Off!)
            HardReturn()
            HardReturn()
        ENDIF
    ELSE      // using existing doc
        GetWPData(OutputDoc; CurrentDocument!)
        PosDocBottom()
        FontLarge(On!)
        Type("Summariz: appending output to " + OutputDocName + " on ")
        DateText()
        FontLarge(Off!)
        HardReturn()
        HardReturn()
    ENDIF
RETURN


//*************************************************************************
// DoComments@ -- copy over all the comments from the current doc to the
output doc.
LABEL(DoComments@)
    GotoNum := InitialCurDoc
    CALL(GotoDoc@)
    PosDocTop()
    // The only exit from this loop is via the OnNotFound trap
    WHILE (TRUE)
        CALL(GetComment@)
        CALL(CopyComment@)
    ENDWHILE
    // Target for local not-found handler
    // LABEL(DoCommentsEnd@)
RETURN


//*************************************************************************
```

```
// GetComment@ -- find the next comment in the current document, and load
it into
// the clipboard; load the source docname and page number into
StoredNamed
// and StoredPage.
LABEL(GetComment@)
    // find comment
    SearchText(SearchString:""; SearchDirection:Forward!;
SearchScope:Extended!)
    GetWPData(SPTemp; Page!)
    NUMSTR(StoredPage; 0; SPTemp)
    GetWPData(StoredName; Name!)
    // Move back to before the comment
    PosCharPrevious()
    // Start selecting
    SelectMode(On!)
    // Move forward over the comment
    PosCharNext()
    // Grab the comment to the clipboard
    EditCopy()
    SelectMode(Off!)
RETURN


//*************************************************************************
// CopyComment@ -- copy the contents of the comment in the clipboard to
the doc
//       with number OutputDoc, without altering what doc is current
LABEL(CopyComment@)
    GetWPData(LocalCurDoc; CurrentDocument!)
    GotoNum := OutputDoc
    CALL(GotoDoc@)
    ParagraphHangingIndent()
    FontRedline(On!)
    Type("Comment from " + StoredName + " page " + StoredPage + ": ")
    FontRedline(Off!)
    EditPaste()
    CommentConvertToText()
    PosDocBottom()
    HardReturn()
    HardReturn()
    GotoNum := LocalCurDoc
    CALL(GotoDoc@)
RETURN
```

```
//*************************************************************************
// DoFlaggedText@ -- copy over all the pieces of flagged text from the current doc to the
//       output doc.
LABEL(DoFlaggedText@)
    GotoNum := InitialCurDoc
    CALL(GotoDoc@)
    PosDocTop()
    // The only exit from this loop is via the OnNotFound trap
    WHILE (TRUE)
        CALL(GetFlaggedString@)
        CALL(CopyFlaggedString@)
    ENDWHILE
RETURN


//*************************************************************************
// GetFlaggedString@ -- find the next flagged string in the current document, and load it into
// the clipboard; load the source docname and page number into the vars StoredNamed
// and StoredPage.
LABEL(GetFlaggedString@)
    // find start of flagged section
    SearchText(SearchString:FlagOpen; SearchDirection:Forward!; SearchScope:Extended!)
    GetWPData(SPTemp; Page!)
    NUMSTR(StoredPage; 0; SPTemp)
    GetWPData(StoredName; Name!)
    // start selecting
    SelectMode(On!)
    // find end of flagged section
    SearchText(SearchString:FlagClose; SearchDirection:Forward!; SearchScope:Extended!)
    // exclude the CloseFlag itself from the selection
    FOR(i; 0; i<=FlagClLen; i+1)
        PosCharPrevious()
    ENDFOR
    EditCopy()
    SelectMode(Off!)
RETURN


//*************************************************************************
// CopyFlaggedString@ -- copy the string in the clipboard to the doc with
//       number OutputDoc, without altering what doc is current
```

```
LABEL(CopyFlaggedString@)
    GetWPData(LocalCurDoc; CurrentDocument!)
    GotoNum := OutputDoc
    CALL(GotoDoc@)
    ParagraphHangingIndent()
    FontRedline(On!)
    Type("Flagged text from " + StoredName + " page " + StoredPage + ": ")
    FontRedline(Off!)
    EditPaste()
    HardReturn()
    HardReturn()
    GotoNum := LocalCurDoc
    CALL(GotoDoc@)
RETURN


//*************************************************************************
// GotoDoc@ -- go to the document whose number is stored in the var
GotoNum
LABEL(GotoDoc@)
    REPEAT
        DocNext()
        GetWPData(CurDoc; CurrentDocument!)
    UNTIL (CurDoc = GotoNum)
RETURN


//*************************************************************************
// QQuit@ -- clean up and exit
LABEL(QQuit@)
    IF (DocSum)
        PrefDocSummary(CreateOnExit:Yes!)
    ENDIF
    QUIT
```