```
'Copyright 2011, Clyde Wilson
'Version 1.2

'Please run this code and change it if necessary.
'Let me know of any requests or changes and I'll
'incorporate them in the next version.

'Formatter runs in Liberty Basic or Just Basic
'I like to use the formatter when tokenized and executed from the run
menu!
'In the Setup menu, choose Preferences and check Reload File On Activa
te
'to get the quickest feedback!
'The above is not necessary, just more fun...
'Be sure to save your code before running formatter
'otherwise you will lose any unsaved changes.
'Formatter indents code blocks and continuation lines and inserts blan
k lines
'around subs and functions. Blank lines inserted by the user are maint
ained.


nomainwin
dim strtBlock$(100)
dim endBlock$(100)
dim alignLeft$(100)
dim info$(10, 10) 'for file check

tabSize = 4
insLines = 1 'For Debug, set to 1 if blank lines are to be inserted
remExcess = 0
'For Debug, set to 1 if excess blank lines are to be removed
blankLine=1

on error goto [error]
gosub [fillArrays]
gosub [openFiles]

[loop]
do
if eof(#text) <> 0 then [quit]
line input #text, item$
wrkLine$ = trim$(item$)
if remExcess = 0 then exit do
loop while len(wrkLine$) = 0
' or (wrkLine$="'") 'to remove single quote lines.
```

```
firstWord$ = lower$( word$( wrkLine$, 1 ))
gosub [chkExepts]
gosub [chkKeys]
gosub [wrtLine]
goto [loop]

[quit]
close #text
close #newText

[contErr]
kill fName$ + " - bkp.bas"
name fName$ + ".bas" as fName$ + " - bkp.bas"
name "formatterTempFile" as fName$ + ".bas"
notice "Finished"
end

[error]
if Err = 0 then
open fName$ + " - bkp.bas" for output as #temp
close #temp
goto [contErr]
else
noteStg$ = "You have received error #"; Err
notice "Error" + chr$(13) + noteStg$ + chr$(13) + Err$ + chr$(13) +
 fName$
close #newText
kill "formatterTempFile"
end
end if

[fillArrays]
counter = 1
do until choice$ = "end"
read choice$
if choice$ <> "end" then
strtBlock$(counter) = choice$
aryCtr = aryCtr + 1
counter = counter + 1
end if
loop
counter = 1
choice$ = ""
do until choice$ = "end"
read choice$
if choice$ <> "end" then
```

```
endBlock$(counter) = choice$
ary2Ctr = ary2Ctr + 1
counter = counter + 1
end if
loop
counter = 1
choice$ = ""
do until choice$ = "end"
read choice$
if choice$ <> "end" then
alignLeft$(counter) = choice$
ary3Ctr = ary3Ctr + 1
counter = counter + 1
end if
loop
return

[openFiles]
confirm "Have you saved your current work?" ; ans$
if ans$="no" then
notice "Please save your current work before proceding"
end
end if
files DefaultDir$, "formatterTempFile", info$()
If val(info$(0, 0)) > 0 then 'the file exists
notice "Please rename or delete formatterTempFile"
end
else
open "formatterTempFile" for output as #newText
end if
prompt "Please type a filename (without the extension)"; fName$
if fName$="" then
notice "Program Not Run - Please enter a filename"
close #newText
kill "formatterTempFile"
end
end if
if lower$(fName$)="untitled" then
notice "Program must be saved under another name before continuing"
close #newText
kill "formatterTempFile"
end
end if
open fName$ + ".bas" for input as #text
notice "Your original file will be found at ";fName$ + " - bkp.bas"
numTabs = 1
```

```
return

[chkExepts] 'Check exceptional cases
indentOk = 1
somethingHere = 0
wrkLineL$ = lower$( wrkLine$ )
secondWord$ = word$( wrkLineL$, 2 )
if firstWord$ = "end" then firstWord$ = firstWord$ + " " + secondWord$
if firstWord$ = "if" then
thenPos = instr( wrkLineL$, " then ")
if thenPos <> 0 then
thenPos = thenPos + 6
for counter = thenPos to len( wrkLineL$ )
if mid$( wrkLineL$, counter, 1 ) <> " " then
somethingHere = 1
exit for
end if
next
end if
if somethingHere then
if mid$(wrkLineL$, counter, 1) ="'" then
indentOk = 1
else
indentOk = 0
end if
end if
end if
return

[chkKeys]
indent = 0
outdent = 0
saveTabs = 0
for counter = 1 to aryCtr
if strtBlock$( counter ) = firstWord$ then
indent = 1
end if
next
for counter = 1 to ary2Ctr
if endBlock$( counter ) = firstWord$ then outdent = 1
next
for counter = 1 to ary3Ctr
if alignLeft$(counter) = firstWord$ then
if numTabs > 1 then saveTabs = numTabs - 1
numTabs = 0
indent = 1
```

```
end if
next
if insLines and needBlanks and blankLine=0 then
if len( wrkLine$)<>0 then print #newText, ""
needBlanks=0
blankLine=1
end if
needBlanks=0
if instr( firstWord$, "[" ) = 1 then
if numTabs > 1 then saveTabs = numTabs - 1
numTabs = 0
indent = 1
if insLines and blankLine=0 then
print #newText, ""
blankLine=1
end if
end if
if firstWord$="sub" or firstWord$="function" then
if insLines and blankLine=0 then
print #newText, ""
blankLine=1
end if
end if
if insLines and (firstWord$="end sub" or firstWord$=
"end function") then needBlanks=1
return

[wrtLine]
if indent then
if firstWord$ = "case" then
if notFirstCase then
numTabs = numTabs - 1
else
notFirstCase = 1
end if
end if
gosub [prtWithTabs]
if saveTabs > 0 then numTabs = saveTabs
if indentOk then numTabs = numTabs + 1
return
end if
if outdent then
numTabs = numTabs - 1
gosub [prtWithTabs]
if firstWord$ = "end select" then
numTabs = numTabs - 1
```

```
notFirstCase = 0
end if
if firstWord$ = "else" then numTabs = numTabs + 1
return
end if
gosub [prtWithTabs]
gosub [checkCont]
return

[prtWithTabs]
' if insLines=0 or len( wrkLine$)<>0 or remExcess=0 or blankLine=0 the
n
for counter = 1 to numTabs
print #newText, space$(tabSize);
next
print #newText, wrkLine$
' end if
gosub [checkBlank]
return

[checkCont]
indentOk = 1
somethingHere = 0
if instr(wrkLineL$, "_") then
endPos = instr( wrkLineL$, "_")
endPos = endPos + 1
for counter = endPos to len( wrkLineL$ )
if mid$( wrkLineL$, counter, 1 ) <> " " then
somethingHere = 1
exit for
end if
next
if somethingHere then
if mid$(wrkLineL$, counter, 1) ="'" then
indentOk = 1
else
indentOk = 0
end if
end if
end if
if instr(wrkLineL$, "_") then
if indentOk then
if firstCont = 0 then
firstCont = 1
numTabs = numTabs + 1
end if
```

```
end if
else
if firstCont then
numTabs = numTabs - 1
firstCont = 0
end if
end if
return

[checkBlank]
if insLines and len( wrkLine$) = 0 then
blankLine=1
else
blankLine=0
end if
return

'Data to start a block
data "case"
data "do"
data "for"
data "function"
data "if"
data "select"
data "sub"
data "while"
data "end"

'Data to end a block
data "else"
data "end function"
data "end if"
data "end select"
data "end sub"
data "loop"
data "next"
data "wend"
data "end"

'Data to align to left edge
data "end function"
data "end sub"
data "function"
data "sub"
data "end"
```