Sample input

```
now is the time for all
good men and women to come to
the aid of their country
```

Sample output

```
                  the aid of their country
  now is the time for all
good men and women to come to
     the aid of their country
                      good men and women to come to
                  now is the time for all
                 good men and women to come to
                      now is the time for all
           the aid of their country
           now is the time for all
         good men and women to come to
```

```
########################################################################
˝
#
˝
#     File:     kwic.icn
˝
#
˝
#     Subject:  Program to produce keywords in context
˝
#
˝
#     Author:   Stephen B. Wampler, modified by Ralph E. Griswold
#
#     Date:     February 15, 1995
#
########################################################################
#
#     <munch, including info on this being public domain>
#
#     Some noise words are omitted (see "exceptions" in the program
#  text).  If a file named except.wrd is open and readable in the
#  current directory, the words in it are used instead.
#
#     This program is pretty simple.  Possible extensions include ways
#  of specifying words to be omitted, more flexible output formatting,
#  and so on.  Another "embellisher's delight".
#
########################################################################
```

```
global line, loc, exceptions, width
˝

˝
procedure main(args)
˝
   local exceptfile
˝

˝
   width := integer(args[1]) | 40
   if exceptfile := open("except.wrd") then {
      exceptions := set()
      every insert(exceptions, lcword(exceptfile))
      close(exceptfile)
      }
   else
      exceptions := set(["or", "in", "the", "to", "of", "on", "a",
         "an", "at", "and", "i", "it", "by", "for"])
   every write(kwic(&input))
end

procedure kwic(file)
   local index, word

#  Each word, in lowercase form, is a key in the table "index".
#  The corresponding values are lists of the positioned lines
#  for that word.  This method may use an impractically large
#  amount of space for large input files.

   index := table()
   every word := lcword(file) do {
      if not member(exceptions,word) then {
         /index[word] := []
         index[word] := put(index[word],position())
         }
      }

#  Before the new sort options, it was done this way – the
#  code preserved as an example of "generators in action".
#  suspend !((!sort(index,1))[2])

   index := sort(index,3)
   while get(index) do
      suspend !get(index)
end

procedure lcword(file)
   static chars
   initial chars := &ucase ++ &lcase ++ &digits ++ '\''
   every line := !file do
      line ? while tab(loc := upto(chars)) do
         suspend map(tab(many(chars)) \ 1)
end

procedure position()
   local offset  #  Note that "line" and ""loc" are global.
   offset := width - loc
   if offset >= 0 then return repl(" ",offset) || line
   else return line[-offset + 1:0]
end
```