The word appearance creation problem:
Given one line plus either the "j" value from a previous word appearance (with that line) or j=0, FIND the next word at or after j if there is any.
Algorithm: Finite State Automaton (or scanner).

Inputs:

\[
\text{pch} \quad \text{One line:} \quad \text{word; } /0
\]

initial position
(0 or previous j)

Variables:

\[
\text{const char * pline = pch;}
\]

(const char means we won’t change the chars in the array)

\[
\text{int c = initial position} \quad \text{//locates NEXT char to process}
\]

\[
\text{enum state_t state = Ready;}
\]

\[
\text{int i, j;} \quad \text{// to report position of found word.}
\]

Reference: Enumeration type (Strou. sec 4.8)
Initialize: current char. position; and CURRENT STATE=“Start State”

Finite state scanner loop:

1. Retrieve and classify the current character.

2. Depending on the CURRENT STATE and CURRENT CHAR. CLASS:
   (a) Possibly take an action.
   (b) Choose a new state (may be the same as current).

3. Exit if new state is “DONE”; otherwise make CURRENT STATE be the new state and make the current character be the next.

#include "classify.h"

enum state_t { Ready, Not, Inside, DONE };  // current STATE
state_t state;  // NEXT char
int c;          // position of NEXT char to analyze.
chclass_t chclass;  // class of NEXT char
const char *pline = ...; // line to scan
int i;          // i will record the index where the
                //   word being found starts,
                //   i==-1 to report none found.
int j;          // j will record the index just
                //   after the word’s end.

// Code in a loop that prepares for scanning
i = -1;
state = Ready;
c = ...;  // either 0 or previous value of j
// classify.h Classifier for kwic333
enum chclass_t { Space, Letter, Other, Null }; extern chclass_t classify( char ch );

// classify.cpp Classifier for kwic333
#include <ctype.h>
#include "classify.h"
chclass_t classify( char ch )
{
    if( ch == ' ' ) return Space;
    if( isalpha( ch ) ) return Letter;
    if( ch == '\0' ) return Null;
    return Other;
}
while (state != DONE) {
  chclass = classify( pline[c] );
  switch( state ) {
    case Ready:
      switch( chclass ) {
        case Space:
          c++; state = Ready; break;
        case Other:
          c++; state = Not; break;
        case Letter:
          i = c;
          c++; state = Inside; break;
        case Null:
          // You figure out!!
      }
      break; // end of state==Ready case

    case Inside: //in switch(state) block

case Inside:  // in switch(state) block
    switch(chclass) {
        case Space:
            // You figure out!!
            // more chclass cases
    }
    break;  // end of state==Inside case

    case Not:  // in switch(state) block
        // Handle state==Not chclass cases.
        // More state cases.
    } // end of switch( state )
} // end of while
if( i < 0 )
{
    // finish with this line
}
// Create a new word appearance
// Continue a loop to seek another.