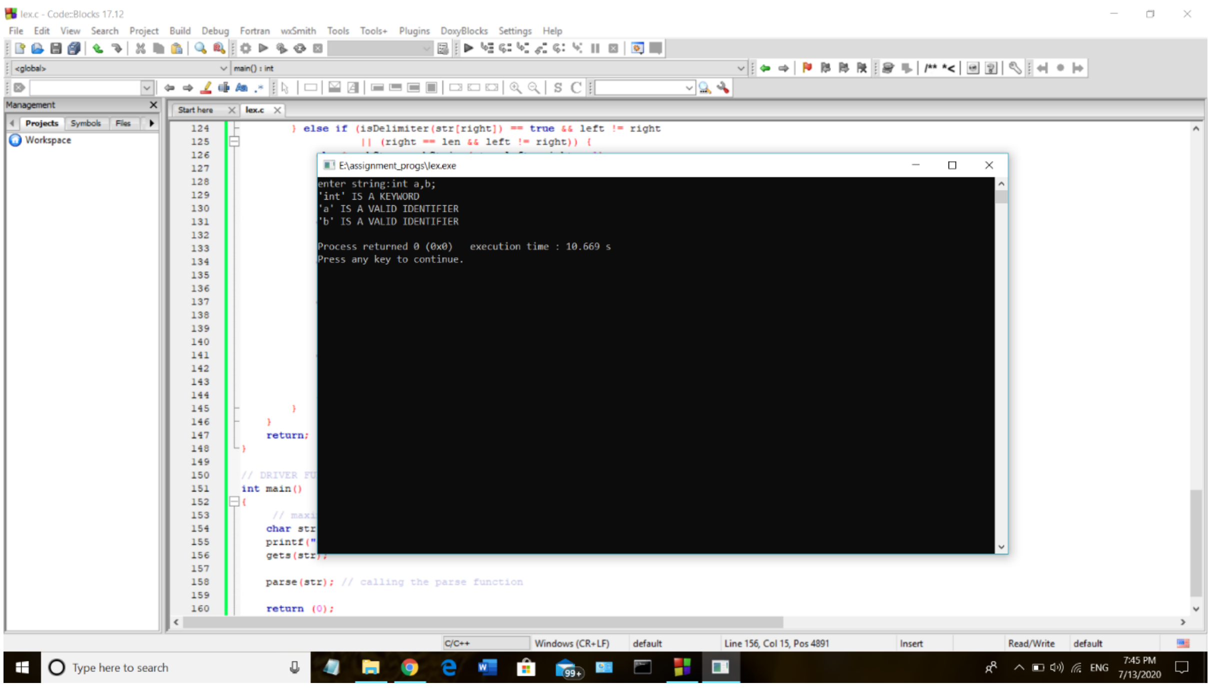
```
#include <stdbool.h>

#include <stdio.h>

#include <string.h>

#include <stdlib.h>


// Returns 'true' if the character is a DELIMITER.

bool isDelimiter(char ch)

{

   if (ch == ' ' || ch == '+' || ch == '-' || ch == '*' ||

       ch == '/' || ch == ',' || ch == ';' || ch == '>' ||

       ch == '<' || ch == '=' || ch == '(' || ch == ')' ||

       ch == '[' || ch == ']' || ch == '{' || ch == '}')

       return (true);

   return (false);

}


// Returns 'true' if the character is an OPERATOR.

bool isOperator(char ch)

{

   if (ch == '+' || ch == '-' || ch == '*' ||

       ch == '/' || ch == '>' || ch == '<' ||

       ch == '=')

       return (true);

   return (false);

}


// Returns 'true' if the string is a VALID IDENTIFIER.

bool validIdentifier(char* str)

{

   if (str[0] == '0' || str[0] == '1' || str[0] == '2' ||

       str[0] == '3' || str[0] == '4' || str[0] == '5' ||
       str[0] == '6' || str[0] == '7' || str[0] == '8' ||
       str[0] == '9' || isDelimiter(str[0]) == true)
       return (false);
   return (true);
}

// Returns 'true' if the string is a KEYWORD.
bool isKeyword(char* str)
{
   if (!strcmp(str, "if") || !strcmp(str, "else") ||
       !strcmp(str, "while") || !strcmp(str, "do") ||
       !strcmp(str, "break") ||
        !strcmp(str, "continue") || !strcmp(str, "int")
       || !strcmp(str, "double") || !strcmp(str, "float")
       || !strcmp(str, "return") || !strcmp(str, "char")
       || !strcmp(str, "case") || !strcmp(str, "char")
       || !strcmp(str, "sizeof") || !strcmp(str, "long")
       || !strcmp(str, "short") || !strcmp(str, "typedef")
       || !strcmp(str, "switch") || !strcmp(str, "unsigned")
       || !strcmp(str, "void") || !strcmp(str, "static")
       || !strcmp(str, "struct") || !strcmp(str, "goto"))
       return (true);
   return (false);
}

// Returns 'true' if the string is an INTEGER.
bool isInteger(char* str)
{
   int i, len = strlen(str);

   if (len == 0)
```

```
        return (false);
    for (i = 0; i < len; i++) {
        if (str[i] != '0' && str[i] != '1' && str[i] != '2'
            && str[i] != '3' && str[i] != '4' && str[i] != '5'
            && str[i] != '6' && str[i] != '7' && str[i] != '8'
            && str[i] != '9' || (str[i] == '-' && i > 0))
            return (false);
    }
    return (true);
}

// Returns 'true' if the string is a REAL NUMBER.
bool isRealNumber(char* str)
{
    int i, len = strlen(str);
    bool hasDecimal = false;

    if (len == 0)
        return (false);
    for (i = 0; i < len; i++) {
        if (str[i] != '0' && str[i] != '1' && str[i] != '2'
            && str[i] != '3' && str[i] != '4' && str[i] != '5'
            && str[i] != '6' && str[i] != '7' && str[i] != '8'
            && str[i] != '9' && str[i] != '.' ||
            (str[i] == '-' && i > 0))
            return (false);
        if (str[i] == '.')
            hasDecimal = true;
    }
    return (hasDecimal);
}
```

```
// Extracts the SUBSTRING.
char* subString(char* str, int left, int right)
{
   int i;
   char* subStr = (char*)malloc(
              sizeof(char) * (right - left + 2));

   for (i = left; i <= right; i++)
      subStr[i - left] = str[i];
   subStr[right - left + 1] = '\0';
   return (subStr);
}

// Parsing the input STRING.
void parse(char* str)
{
   int left = 0, right = 0;
   int len = strlen(str);

   while (right <= len && left <= right) {
      if (isDelimiter(str[right]) == false)
         right++;

      if (isDelimiter(str[right]) == true && left == right) {
         if (isOperator(str[right]) == true)
            printf("'%c' IS AN OPERATOR\n", str[right]);

         right++;
         left = right;
      } else if (isDelimiter(str[right]) == true && left != right
                || (right == len && left != right)) {
            char* subStr = subString(str, left, right - 1);

            if (isKeyword(subStr) == true)
                printf("'%s' IS A KEYWORD\n", subStr);

            else if (isInteger(subStr) == true)
                printf("'%s' IS AN INTEGER\n", subStr);

            else if (isRealNumber(subStr) == true)
                printf("'%s' IS A REAL NUMBER\n", subStr);

            else if (validIdentifier(subStr) == true
                    && isDelimiter(str[right - 1]) == false)
                printf("'%s' IS A VALID IDENTIFIER\n", subStr);

            else if (validIdentifier(subStr) == false
                    && isDelimiter(str[right - 1]) == false)
                printf("'%s' IS NOT A VALID IDENTIFIER\n", subStr);
            left = right;
        }
    }
    return;
}

// DRIVER FUNCTION
int main()
{
     // maximum legth of string is 100 here
    char str[100];
    printf("enter string:");
    gets(str);

    parse(str); // calling the parse function

    return (0);
}
```

Output:



```
            } else if (isDelimiter(str[right]) == true && left != right
                    || (right == len && left != right)) {
```

E:\assignment_progs\lex.exe

```
enter string:int a,b;
'int' IS A KEYWORD
'a' IS A VALID IDENTIFIER
'b' IS A VALID IDENTIFIER

Process returned 0 (0x0)   execution time : 10.669 s
Press any key to continue.
```

```
        parse(str); // calling the parse function

        return (0);
```

lex.c - Code::Blocks 17.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Symbols Files
Workspace
Start here
lex.c
C/C++
Windows (CR+LF)
default
Line 156, Col 15, Pos 4891
Insert
Read/Write
Type here to search
7:45 PM
7/13/2020