```c
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX_BUFFER 1000

bool isDelimiter(char ch)
{
    if (ch == ' ' || ch == '+' || ch == '-' || ch == '*' ||
        ch == '/' || ch == ',' || ch == ';' || ch == '>' ||
        ch == '<' || ch == '=' || ch == '(' || ch == ')' ||
        ch == '[' || ch == ']' || ch == '{' || ch == '}' || ch == '\n' || ch == '\t' ||
ch=='\0' )
        return (true);
    return (false);
}

bool isOperator(char ch)
{
    if (ch == '+' || ch == '-' || ch == '*' ||
        ch == '/' || ch == '>' || ch == '<' ||
        ch == '=')
        return (true);
    return (false);
}

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);
}

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);
}

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);
}
```

```
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);
}

void removeComments(char* str)
{
    int i, j;
    int len = strlen(str);
    char buffer[MAX_BUFFER];
    bool inSingleLineComment = false;
    bool inMultiLineComment = false;
    for (i = 0, j = 0; i < len; i++)
    {
        if (inSingleLineComment)
        {
            if (str[i] == '\n') {
            inSingleLineComment = false;
            buffer[j++] = str[i];
            }
        }
        else if (inMultiLineComment) {
            if (str[i] == '*' && i + 1 < len && str[i + 1] == '/')
            {
                inMultiLineComment = false;
                i++; // Skip '/'
            }
        }
        else
        {
            if (str[i] == '/' && i + 1 < len)
            {
                if (str[i + 1] == '/') {
                    inSingleLineComment = true;
                    i++; // Skip '/'
                }
                else if (str[i + 1] == '*') {
                    inMultiLineComment = true;
                    i++; // Skip '*'
                }
                else
                    buffer[j++] = str[i];
            }
            else
                buffer[j++] = str[i];
        }
    }
    // Null-terminate the result and copy it back
    buffer[j] = '\0';
    strcpy(str, buffer);
}
```

```
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);
}

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;
}

int main()
{
    char str[100],temp[100];
    FILE* fptr;
    fptr = fopen("input.txt","r");
    bzero(str,100);
    while(!feof(fptr))
    {
        bzero(temp,100);
        fgets(temp,100,fptr);
        strcat(str,temp);
    }
    printf("%s\n",str);
    removeComments(str);
    parse(str);
    return (0);
}
```