North South University
Department of Electrical & Computer Engineering

Homework1

Tokenize the C statement and identify C keyword

Course Code: CSE425
Section: 08
Faculty Initial: MdAR
Student Name: MD Sadikul Haque Sadik
ID: 2031093642

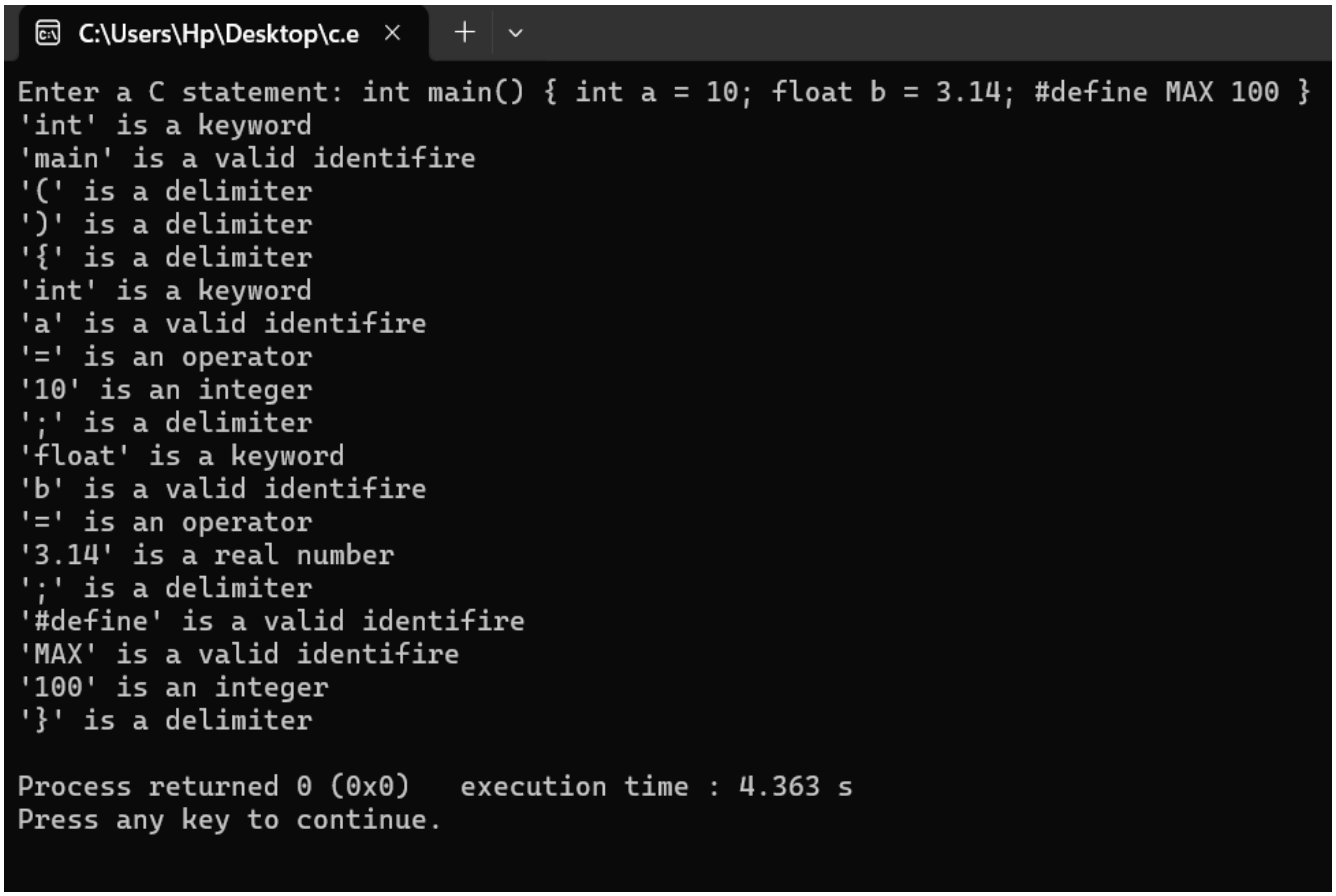
C Code for Tokenizing a C Statement

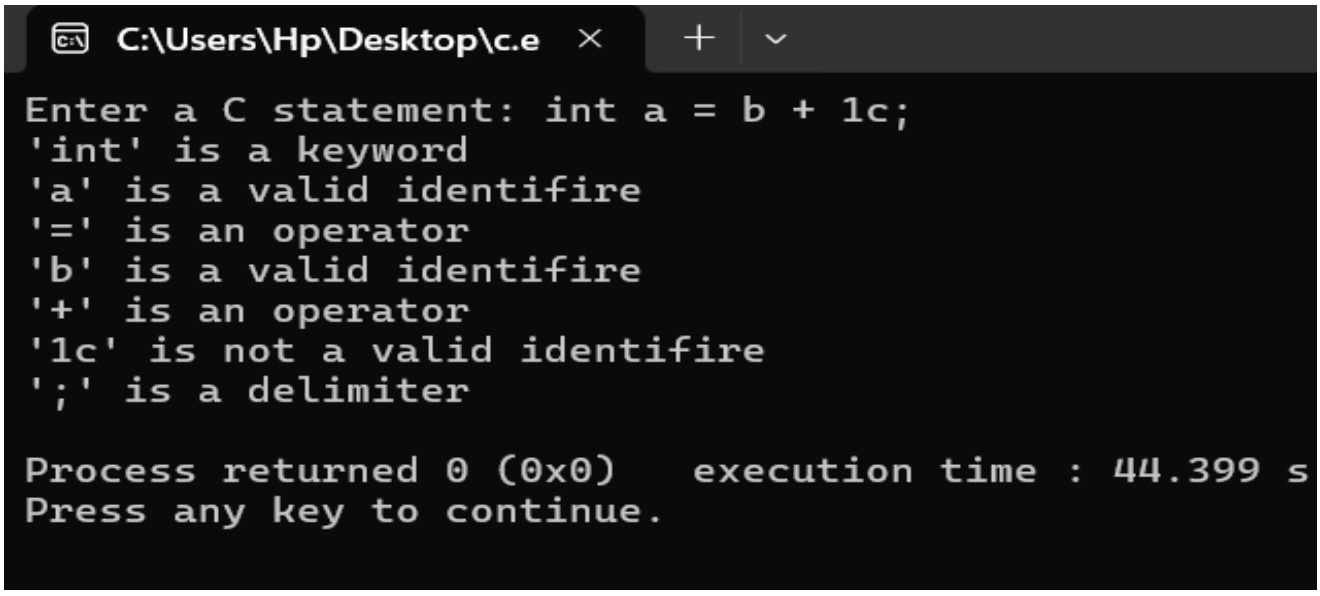
```
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#define MAX_LENGTH 100

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);
}

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);
}
```

```
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 detectTokens(char* str) {
   int left = 0, right = 0;
   int len = strlen(str);

   while (right <= len && left <= right) {
      if (!isDelimiter(str[right]) && str[right] != '\n')
         right++;

      if ((isDelimiter(str[right]) && left == right) || str[right] ==
'\n') {
         if (isOperator(str[right]))
            printf("'%c' is an operator\n", str[right]);
         else if (isDelimiter(str[right]) && str[right] != ' ' &&
str[right] != '\n')
            printf("'%c' is a delimiter\n", str[right]);
         right++;
         left = right;
      } else if ((isDelimiter(str[right]) && left != right) || (right
== len && left != right)) {
         char* subStr = subString(str, left, right - 1);

         if (isKeyword(subStr))
            printf("'%s' is a keyword\n", subStr);
         else if (isInteger(subStr))
            printf("'%s' is an integer\n", subStr);
         else if (isRealNumber(subStr))
            printf("'%s' is a real number\n", subStr);
         else if (validIdentifier(subStr) && !isDelimiter(str[right -
1]))
            printf("'%s' is a valid identifire\n", subStr);
         else if (!validIdentifier(subStr) && !isDelimiter(str[right
- 1]))
            printf("'%s' is not a valid identifire\n", subStr);

         free(subStr);
         left = right;
      }
   }
}
```

```
int main() {
   char str[MAX_LENGTH];

   printf("Enter a C statement: ");
   fgets(str, MAX_LENGTH, stdin);

   detectTokens(str);

   return 0;
}
```

Sample Input and Output





Here's a summary table of lexeme categories based on my C code. This table aligns with the functionality of detectTokens, showing which categories are detected and examples relevant to the code.

Category	**Examples**
Keywords	int, return, if, while, float, double, else, do, void
Identifiers	variable, myFunction, count
Operators	+, -, *, /, =, >, <, ==
Literals	123, 3.14, 'c', "Hello World"
Delimiters	{, }, (,), [,], ;, ,
Invalid Identifiers	1variable, 9name
Integers	42, 1000, -7
Real Numbers	3.14, 0.001, -12.5
Unknown Tokens	Any token not matching the above