Exp No: 1

Date:

IMPLEMENT CODE TO RECOGNIZE TOKENS IN C

AIM:

To implement the program to identify C keywords, identifiers, operators, end statements like [], {} using C tool.

ALGORITHM:

1. Start
2. Define functions to check if a character is a delimiter, operator, or a valid identifier.
3. Define functions to check if a given string is a keyword, integer, real number, or a valid identifier based on certain conditions.
4. Define a function to extract substrings from the input string based on delimiter positions.
5. Define a parsing function that iterates through the input string character by character and identify substrings delimited by spaces or operators.
6. Check each substring for being a keyword, integer, real number, or a valid identifier and print the corresponding message.
7. Define the main function.
8. Initialize a string with the input expression.
9. Call the parsing function with the input string.
10. Print the results of the parsing, indicating whether substrings are keywords, integers, real numbers, or valid identifiers.

Roll Number: 210701067
Name: HARI AMERTHESH N

PROGRAM:

```
#include <stdbool.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
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)
```

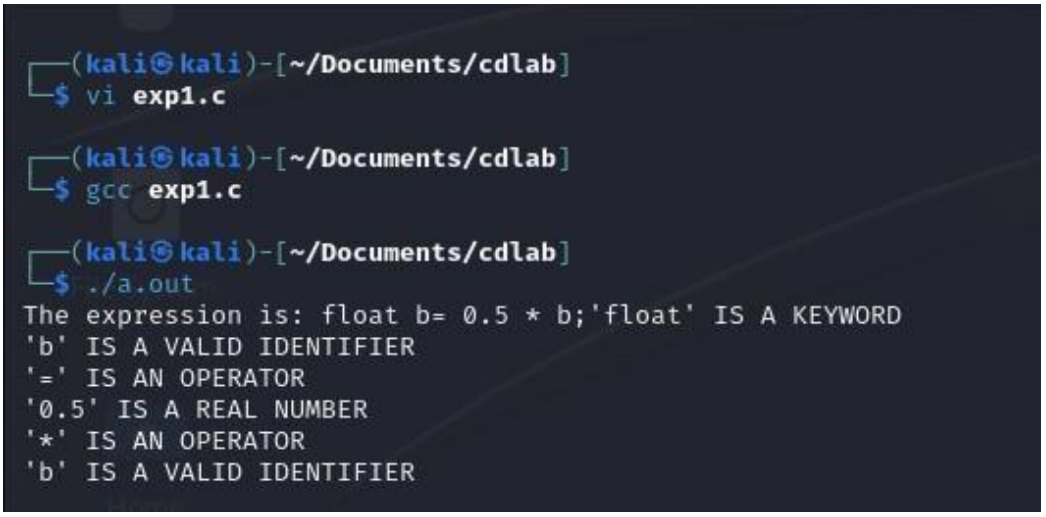
```
                return (false);
        for (i = 0; i < len; i++) {
                if (str[i] != '0' && str[i] != '1' && str[i] != '2'
                        && str[i] != '3' && str[i] != '4' && str[i] != '5'
                        && str[i] != '6' && str[i] != '7' && str[i] != '8'
                        && 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 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()
{
        // maximum length of string is 100 here
        printf("The expression is: float b= 0.5 * b;\n");
        char str[100] = "float b = 0.5 * b; ";
```

Roll Number: 210701067
Name: HARI AMERTHESH N

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


    return (0);
}
```

OUTPUT:



RESULT:

Thus, a C program is implemented to identify C keywords, identifiers, operators and end statements.