VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

B.Tech. Winter Semester 2023-24

School Of Electronics Engineering

(SENSE)

COMPILER DESIGN
BCSE307P

LAB Experiment - 1

SUJAY GHOSH

21BLC1607

AIM: Write a C program to detect tokens for Lexical Analyzer

C - 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);
```

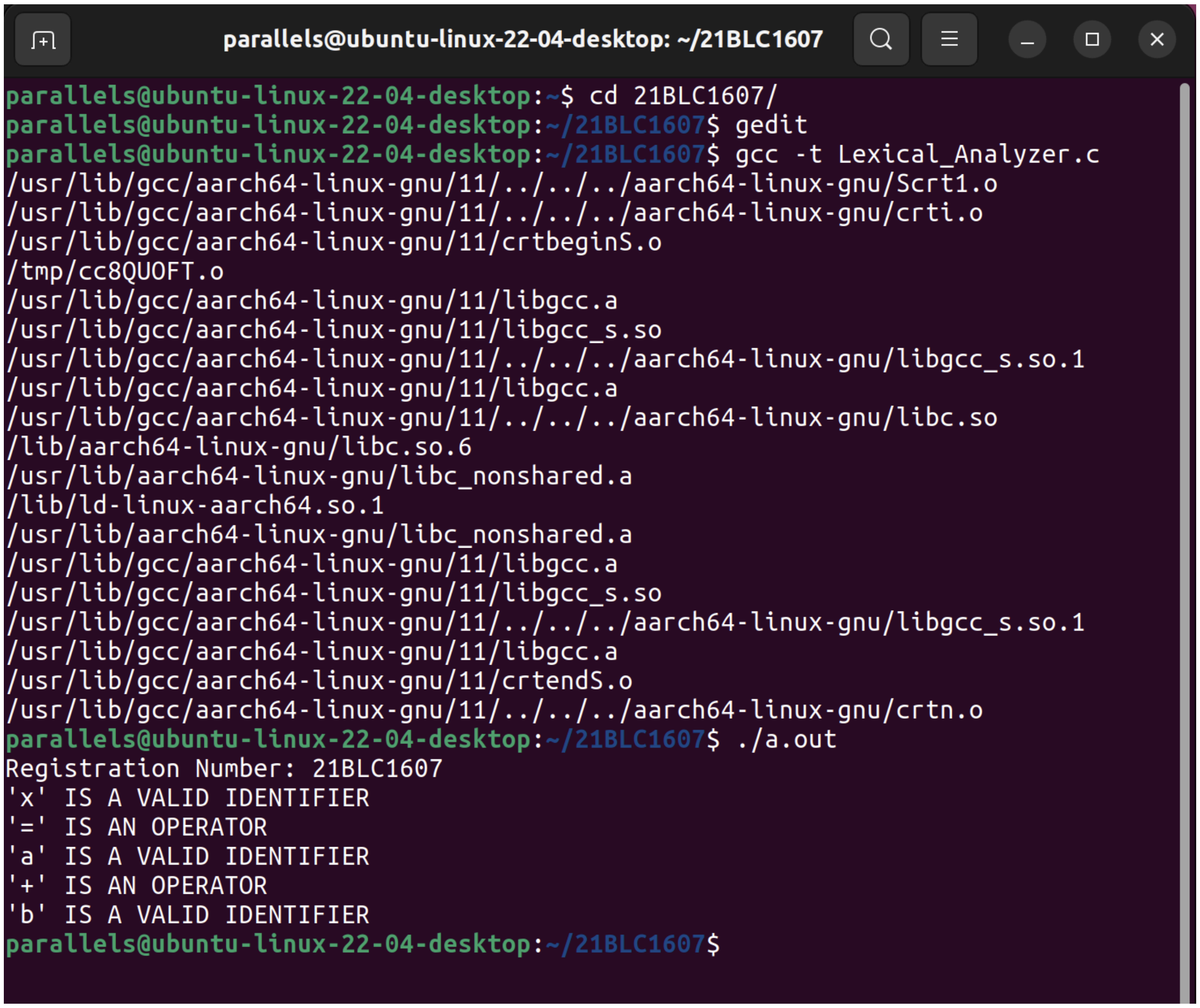
```
		if (str[i] == '.')
			hasDecimal = true;
	}
	return (hasDecimal);
}

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()
{
        printf("Registration Number: 21BLC1607\n");
        char str[100] = "x = a + b; ";
        parse(str);
        return (0);
}
```

Output -



parallels@ubuntu-linux-22-04-desktop: ~/21BLC1607

```
parallels@ubuntu-linux-22-04-desktop:~$ cd 21BLC1607/
parallels@ubuntu-linux-22-04-desktop:~/21BLC1607$ gedit
parallels@ubuntu-linux-22-04-desktop:~/21BLC1607$ gcc -t Lexical_Analyzer.c
/usr/lib/gcc/aarch64-linux-gnu/11/../../../aarch64-linux-gnu/Scrt1.o
/usr/lib/gcc/aarch64-linux-gnu/11/../../../aarch64-linux-gnu/crti.o
/usr/lib/gcc/aarch64-linux-gnu/11/crtbeginS.o
/tmp/cc8QUOFT.o
/usr/lib/gcc/aarch64-linux-gnu/11/libgcc.a
/usr/lib/gcc/aarch64-linux-gnu/11/libgcc_s.so
/usr/lib/gcc/aarch64-linux-gnu/11/../../../aarch64-linux-gnu/libgcc_s.so.1
/usr/lib/gcc/aarch64-linux-gnu/11/libgcc.a
/usr/lib/gcc/aarch64-linux-gnu/11/../../../aarch64-linux-gnu/libc.so
/lib/aarch64-linux-gnu/libc.so.6
/usr/lib/aarch64-linux-gnu/libc_nonshared.a
/lib/ld-linux-aarch64.so.1
/usr/lib/aarch64-linux-gnu/libc_nonshared.a
/usr/lib/gcc/aarch64-linux-gnu/11/libgcc.a
/usr/lib/gcc/aarch64-linux-gnu/11/libgcc_s.so
/usr/lib/gcc/aarch64-linux-gnu/11/../../../aarch64-linux-gnu/libgcc_s.so.1
/usr/lib/gcc/aarch64-linux-gnu/11/libgcc.a
/usr/lib/gcc/aarch64-linux-gnu/11/crtendS.o
/usr/lib/gcc/aarch64-linux-gnu/11/../../../aarch64-linux-gnu/crtn.o
parallels@ubuntu-linux-22-04-desktop:~/21BLC1607$ ./a.out
Registration Number: 21BLC1607
'x' IS A VALID IDENTIFIER
'=' IS AN OPERATOR
'a' IS A VALID IDENTIFIER
'+' IS AN OPERATOR
'b' IS A VALID IDENTIFIER
parallels@ubuntu-linux-22-04-desktop:~/21BLC1607$
```