

SSGMCE/FRM/32-B			
SSGMCE	SHRI SANT GAJANAN MAHARAJ COLLEGE OF ENGG.	LABORATORY MANUAL	
	PRACTICAL EXPERIMENT INSTRUCTION SHEET		
	EXPERIMENT TITLE : Write a C program to simulate a lexical analyzer for validating operators.		
EXPERIMENT NO. : SSGMCE/WI/IT/01/6IT01/05		ISSUE NO. : 00	ISSUE DATE : 01.02.2022
REV. DATE :		REV. NO. :	DEPTT. : INFORMATION TECHNOLOGY
LABORATORY : COMPILER DESIGN (CD)		SEMESTER : VI	PAGE: OF 3

**1.0) AIM:**

Write a C program to simulate a lexical analyzer for validating operators.

**2.0) OBJECTIVE:**

After the completion of this experiment, lexical analyzer will be able to validate operators.

**3.0) FACILITIES/ APPARATUS:**

- i) Hardware : Computer Machine
- ii) Software : Turbo C++

**4.0) THEORY:**

The symbols which are used to perform logical and mathematical operations in a C program are called C operators. These C operators join individual constants and variables to form expressions. Operators, functions, constants and variables are combined together to form expressions. Consider the expression  $A + B * 5$ . Where,  $+$ ,  $*$  are operators,  $A$ ,  $B$  are variables,  $5$  is constant and  $A + B * 5$  is an expression.

1. Arithmetic operators
2. Assignment operators
3. Relational operators
4. Logical operators
5. Bit wise operators
6. Conditional operators (ternary operators)
7. Increment/decrement operators
8. Special operators

**5.0) PROGRAM:**

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
void main()
{
    clrscr();
    char arithmetic[5]={'+', '-', '*', '/', '%'};
```



```

char relational[4]={'<','>','!','='};
char bitwise[5]={'&','^','~','|'};
char str[2]={' ',' '};
printf("Enter the operator to identify:");
scanf("%s", &str);
int i;
if(((str[0]=='&' || str[0]=='|') && str[0]==str[1]) || (str[0]=='!' && str[1]=='\0'))
{
    printf("\n It is Logical Operator");
}
for(i=0;i<4;i++)
{
    if(str[0]==relational[i] && (str[1]==' '||str[1]=='\0'))
    {
        printf("\n It is Relational Operator"); break;
    }
}
for(i=0;i<4;i++)
{
    if((str[0]==bitwise[i] && str[1]=='\0') || ((str[0]=='<' || str[0]=='>') && str[1]==str[0]))
    {
        printf("\n It is Bitwise Operator"); break;
    }
}
if(str[0]=='?' && str[1]==':')
printf("\n It is Ternary Operator");
for(i=0;i<5;i++)
{
    if((str[0]=='+' || str[0]=='-') && str[0]==str[1])
    {
        printf("\n It is Unary Operator"); break;
    }
}

```



SHRI SANTI GAJANAN PRAJNAN  
**PRACTICAL EXPERIMENT INSTRUCTION SHEET**

**SSGMCE**

EXPERIMENT TITLE :  
 Write a C program to simulate a lexical analyzer for validating operators.

EXPERIMENT NO. : **SSGMCE/WI/IT/01/6IT01/05**

ISSUE NO. : 00

ISSUE DATE : 01.02.2022

REV. DATE :

REV. NO. :

DEPTT. : INFORMATION TECHNOLOGY

SEMESTER : VI

PAGE: OF 3

LABORATORY : COMPILER DESIGN (CD)

```

    }
    else if((str[0]==arithmetic[i] && str[1]=='=') || (str[0]=='=' && str[1]==' '))
    {
        printf("\n It is Assignment Operator"); break;
    }
    else if(str[0]==arithmetic[i] && str[1]=='\0')
    {
        printf("\n It is Arithmetic Operator"); break;
    }
}
getch();
}
    
```

### 6.0) OUTPUT OF PROGRAM

Enter the operator to identify:

**INPUT**

+

**OUTPUT**

It is Assignment Operator

Enter the operator to identify:

**INPUT**

&&

**OUTPUT**

It is Logical Operator

### 7.0) CONCLUSION:

A lexical analyzer has been designed using C language for the given language in which it verifies the operators.