Date:11/07/25

# EXPERIMENT NO: 1 LEXICAL ANALYSIS USING C

**AIM:** Write a C program to design and implement a lexical analyser to recognize all valid tokens in the input text.The lexical analyser should ignore redundant space,tabs and newlines.

**PROBLEM OBJECTIVE:** To implement lexical analyzer using c program.

**PROBLEM DEFINITION:** A C program that demonstrates working of a lexical analyzer.

## ALGORITHM

1. Start
2. Define keywords,operators and special characters.
3. Open the program file.
4. Read each character from the file.
   1. Check if the character is operator. If yes ,print that it is an operator.
   2. Check if the character is a special character. If yes,print that it is a special character.
   3. Check if the character is an identifier or a keyword. If yes,print.
   4. Check if the character is a constant. If yes,print.
5. Close the file. 6.Stop

## PROGRAM

#include <stdio.h> #include <stdlib.h> #include <string.h> #include <ctype.h>

int isKeyword(char buffer[])

{

char keywords[32][10] = {"auto", "break", "case", "char", "const", "continue",

"default", "do", "double", "else", "enum", "extern", "float", "for", "goto",

"if", "int", "long", "register", "return", "short", "signed", "sizeof",

"static", "struct", "switch", "typedef", "union", "unsigned", "void", "volatile", "while"};

for (int i = 0; i < 32; i++) {

if (strcmp(keywords[i], buffer) == 0) { return 1;

}

}

return 0;

}

int main()

{

char ch, buffer[15], operators[] = "+-\*/%=<>!&|^~", specialch[] = "(){}[];,.:?";

FILE \*fp;

int i, j = 0, k = 0;

fp = fopen("program.txt", "r"); if (fp == NULL)

{

printf("Error while opening the file...\n"); exit(1);

}

while ((ch = fgetc(fp)) != EOF)

{

// Check for operators

for (i = 0; i < strlen(operators); i++) { if (ch == operators[i]) {

printf("%c is an operator\n", ch); break;

}

}

// Check for special characters

for (i = 0; i < strlen(specialch); i++) { if (ch == specialch[i]) {

printf("%c is a special character\n", ch); break;

}

}

// Check for alphabetic characters (identifiers/keywords) if (isalpha(ch)) {

buffer[j++] = ch;

}

else if ((isdigit(ch) || ch == '.') && j > 0) { buffer[j++] = ch;

}

else if (j > 0)

{

buffer[j] = '\0'; j = 0;

if (isKeyword(buffer)) {

printf("%s is a keyword\n", buffer);

}

else {

printf("%s is an identifier\n", buffer);

}

}

// Check for numeric constants if (isdigit(ch)) {

buffer[k++] = ch;

}

else if (ch == '.' && k > 0) {

buffer[k++] = ch;

}

else if (k > 0) { buffer[k] = '\0'; k = 0;

printf("%s is a constant\n", buffer);

}

}

// Handle any remaining buffer content if (j > 0)

{

buffer[j] = '\0';

if (isKeyword(buffer)) {

printf("%s is a keyword\n", buffer);

}

else {

printf("%s is an identifier\n", buffer);

}

}

if (k > 0) {

buffer[k] = '\0';

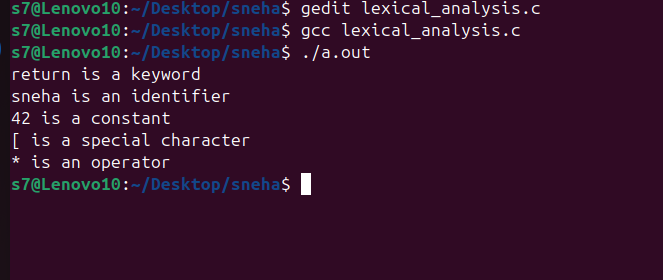
printf("%s is a constant\n", buffer);

}

fclose(fp); return 0;

}

## OUTPUT

****

**RESULT**

The program has been executed successfully and output is obtained.

Date:11/07/25

# EXPERIMENT NO: 2 DFA IMPLEMENTATION

**AIM:** Write a C program to perform DFA implementation.

**PROBLEM OBJECTIVE:** To implement DFA using c program. **PROBLEM DEFINITION:** A C program demonstrating DFA. **ALGORITHM**

1. Start
2. Read the number of states.
3. Read the number of inputs.
4. Read the state..
   1. Check whether final state or not.
5. Read input.
   1. Define transition for each input.
6. Read the input string.
   1. Check whether the character in the string is input.
   2. Check the next state.
7. If it is the final state.
   1. Print the string is accepted.
8. Stop

## PROGRAM

#include<stdio.h> void main()

{

int state[10];

int str[10],input[10]; char ch;

int x[20];

int s,n,k=0,j,a,i,l,t,q=0,fs,b,nxt,z; printf("Enter the no. of states:"); scanf("%d",&s);

printf("Enter the no. of ips:"); scanf("%d",&n); for(i=0;i<s;i++)

{

printf("\nEnter the state %d:",i+1); scanf("%d",&state[i]);

printf("Is final state?..y...1//...n...0:"); scanf("%d",&a);

if(a==1)

{

fs=state[i];

}

}

printf("\nEnter the input:\n"); for(i=0;i<n;i++)

{

scanf("%d",&input[i]);

}

printf("\nTransition state:"); for(i=0;i<s;i++)

{

for(j=0;j<n;j++)

{

printf("\n(q%d,%d)=q",state[i],input[j]); scanf("%d",&b);

x[k]=b; k++;

}

}

do

{

printf("Enter the length of the string:\n"); scanf("%d",&l);

printf("\nEnter the input string:\n"); for(i=0;i<l;i++)

scanf("%d",&str[i]);

q=0;

for(i=0;i<l;i++)

{

t=0;

do

{

if(str[i]==input[t])

{

nxt=x[n\*q+t]; for(j=0;j<s;j++)

{

}

else

} t++;

t++;

if(nxt==state[j]) q=j;

}while(t!=n);

}

if(q==fs)

printf("String Accepted...");

else

printf("String not accepted...");

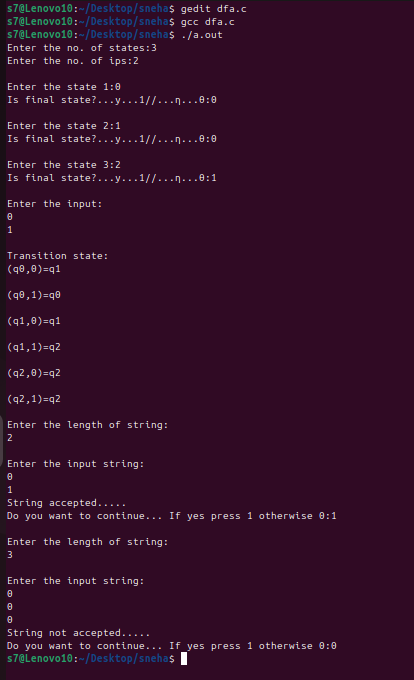
printf("\nDo you want to continue...if yes press 1 otherwise 0:");

scanf("%d",&z);

}while(z==1);

}

## OUTPUT

****

**RESULT**

The program has been executed successfully and output is obtained.

Date:01/08/25

# EXPERIMENT NO: 3

**ε - CLOSURE USING C**

**AIM:** Write a C program to find ε-closure of all states of given NFA with epsilon transition.

**PROBLEM OBJECTIVE:** To find ε-closure using c program.

**PROBLEM DEFINITION:** A C program that demonstrates the computation of ε-closure (epsilon closure) for a given Non-deterministic Finite Automaton (NFA).

## ALGORITHM

1. Start
2. Enter the number of alphabets ,number of states and transitions.
3. For i=0 to i<n of transitions ,insert the transitions to the structure defined .
4. For i=0 to i<n no. of states the transitions find the closure
5. Display the epsilon closure
6. Stop

## PROGRAM

#include <stdio.h> #include <string.h>

char result[20][20], copy[3], states[20][20]; void add\_state(char a[3], int i) { strcpy(result[i], a);

}

void display(int n)

{

int k = 0;

printf("\nEpsilon closure of %s = { ", copy); while (k < n) {

printf("%s ", result[k]); k++;

}

printf("}\n");

}

int main()

{

FILE \*INPUT;

INPUT = fopen("exp3\_input.txt", "r"); char state[3];

int end, i = 0, m, n, k = 0;

char state1[3], input[3], state2[3]; printf("Enter the number of states:\n"); scanf("%d", &n);

printf("Enter the states:\n"); for (k = 0; k < n; k++) { scanf("%s", states[k]);

}

for (k = 0; k < n; k++)

{

i = 0;

strcpy(state, states[k]); strcpy(copy, state); add\_state(state, i++); while (1)

{

end = fscanf(INPUT, "%s %s %s", state1, input, state2); if (end == EOF) {

break;

}

if (strcmp(state, state1) == 0) { if (strcmp(input, "e") == 0) { add\_state(state2, i++); strcpy(state, state2);

}

}

}

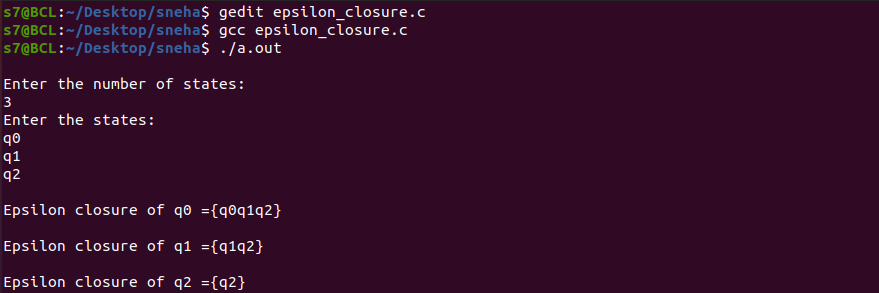
display(i); rewind(INPUT);

}

return 0;

}

## OUTPUT

****

**RESULT**

The program has been executed successfully and output is obtained.

Date:01/08/25

# EXPERIMENT NO:4 RECURSIVE DESCENT PARSING

**AIM:** Write a C program to perform recursive descent parsing.

**PROBLEM OBJECTIVE:** To implement recursive descent parsing using c program.

**PROBLEM DEFINITION:** A C program demonstrating recursive descent parsing.

## ALGORITHM

1. Start
2. Define the variables.
3. Read the arithmetic expression. 4.E()

### 5.T()

4.1.T()

* 1. EPrime()
     1. If input[i] = + then
        1. i=i+1

### 4.2.1.1.T()

* + - 1. EPrime()

### 5.1.F()

5.2.TPrime() 6.TPrime()

* 1. If input[i] = \* then
     1. i=i+1

### 6.1.2.F()

6.1.3.TPrime()

### 7.F()

* 1. If input[i] = ( then
     1. i=i+1

### 7.1.2.E()

7.1.3.If input[i] = ) then set i=i+1

* 1. If isalpha(input([i]) then set i=i+1
  2. Else set error=1

1. If length of input string=i and error=0 then print accepted
2. Else print rejected 10.Stop

## PROGRAM

#include<stdio.h> #include<string.h>

char result[20][20],copy[3],states[20][20]; void add\_state(char a[3],int i)

{

strcpy(result[i],a);

}

void display(int n)

{

int k=0;

printf("\nEpsilon closure of %s ={",copy); while(k<n)

{

printf("%s",result[k]); k++;

}

printf("}\n");

}

int main()

{

FILE \*INPUT;

INPUT=fopen("exp3\_input.txt","r"); char state[3];

int end,i=0,n,k=0;

char state1[3],input[3],state2[3]; printf("\nEnter the number of states:\n"); scanf("%d",&n);

printf("Enter the states:\n"); for(k=0;k<n;k++)

{

scanf("%s",states[k]);

}

for(k=0;k<n;k++)

{

i=0;

strcpy(state,states[k]); strcpy(copy,state); add\_state(state,i++); while(1)

{

end=fscanf(INPUT,"%s%s%s",state1,input,state2); if(end==EOF)

{

break;

}

if(strcmp(state,state1)==0)

{

if(strcmp(input,"e")==0)

{

add\_state(state2,i++); strcpy(state,state2);

}

}

}

display(i); rewind(INPUT);

}

return 0;

}

## OUTPUT



**RESULT**

The program has been executed successfully and output is obtained.