/*Name - Amod Dhopavkar
Roll No - 33304

Lexical Analyzer
*/

Code-->

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

typedef struct UTT
{
	int index;
	char name[100];
	char class[100];
}UTT;

struct UTT UT[100];
char identifiers[30][50];
char literals[40][50];
char trm[20][50];
char keywords[19][10] =
{"break","case","char","const","continue","do","else","float","for","goto","if","int","return","static"
,"struct","switch","typedef","void","while"};
int idindex=0,ltindex=0,trindex=0,ustindex=0;

int isduplit(char tok[30]) {
	int i=0;
	while(i<=ltindex) {
		if(strcmp(tok,literals[i])==0)
		return 1;
		i++;
	}
	return 0;
}

int isdupterminal(char tok[10]) {
	int i=0;
	while(i<=trindex) {
		if(strcmp(tok,trm[i])==0)
		return 1;
		i++;
	}
```

```
        return 0;
}

int isdupidentifier(char tok[10]) {
        int i=0;
        while(i<=idindex) {
                if(strcmp(tok,identifiers[i])==0)
                return 1;
                i++;
        }
        return 0;
}

FILE *sp;
char tok1[20];

int is_keyword(char token[30]) {
        int i=0,flag=0;
        for(i=0;i<19;i++) {
                if(strcmp(token,keywords[i])==0) {
                        flag=1;
                        break;
                }
        }
        return flag;
}

void construct() {
        char buffer[80];
        char token[30];
        sp=fopen("input.txt","r");
        int i=0;
        while(fgets(buffer,80,sp)) {
                int j=0,t=0;
                int len=strlen(buffer);

                while(j<len-1) {
                        //To handle operator
                        if(!isalnum(buffer[j]) && buffer[j]!=' ' && buffer[j]!='_' && buffer[j]!='\t' &&
buffer[j]!='\n') {
                                        char demo[20];
                                        switch(buffer[j]) {
                                        case '\"':
                                        j++;
                                        int ix=0;
                                        while(buffer[j]!='"') {
```

```
                                                  demo[ix]=buffer[j];
                                                  j++;ix++;
                                          }
                                          demo[ix]='\0';
                                          if(isduplit(demo)==0) {
                    strcpy(literals[ltindex],demo);
                    ltindex++;
                  }
                                          break;

                  case '\'':
                                          demo[0]=buffer[++j];
                                          demo[1]='\0';
                                          j++; //   To Skip last single quote
                                          if(isduplit(demo)==0) {
                    strcpy(literals[ltindex],demo);
                    ltindex++;
                  }
                  break;

                  default:
                                            demo[0]=buffer[j];
                                            if((buffer[j]=='+' && buffer[j+1]=='+') || (buffer[j]=='-'
&& buffer[j+1]=='-')) {
                    j++;
                    demo[1]=buffer[j];
                    demo[2]='\0';
                                            }
                                            else
                                              demo[1]='\0';

                  if(isdupterminal(demo)==0) {
                    strcpy(trm[trindex],demo);
                    trindex++;
                                            }
                                          }
                        }

                        if(isalpha(buffer[j])) {
                              int flag=1;
                              while((isalpha(buffer[j]) || buffer[j]=='_' || isdigit(buffer[j])) &&
buffer[j]!=' ' && buffer[j]!='\n' && buffer[j]!='\t') {
                                          flag=0;
                                          token[t++]=buffer[j++];
                              }
```

```
                                    if(flag==0)
                                       j--;
                                    token[t]='\0';

                                    if(is_keyword(token)==1) {
               if(isdupterminal(token)==0) {
                  strcpy(trm[trindex],token);
                  trindex++;
                                             }
                                    }
            else {

                                             if(isdupidentifier(token)==0) {
                  strcpy(identifiers[idindex],token);
                  idindex++;
                                             }
                                    }
                                    memset(token,0,strlen(token));
                                    t=0;

                           }

                           else {
            char demo2[20];
            int di=0;
            if(isdigit(buffer[j])) {
               while(isdigit(buffer[j])) {
                  demo2[di]=buffer[j];
                  di++;j++;
               }
               demo2[di]='\0';
               if(isduplit(demo2)==0) {
                  strcpy(literals[ltindex],demo2);
                  ltindex++;
               }
            }
            }
                           j++;
                  }
                  i++;
         }
         fclose(sp);
}

void check(char tok[30]) {
         int i=0,j=0,k=0;
```

```
        while(i<trindex) {

                if(strcmp(trm[i],tok)==0) {
        strcpy(UT[ustindex].name,tok);
        strcpy(UT[ustindex].class,"Terminal");
        UT[ustindex].index=i;
        ustindex++;
        break;
                }
                i++;
        }

        while(j<ltindex) {
                if(strcmp(literals[j],tok)==0) {
        strcpy(UT[ustindex].name,tok);
        strcpy(UT[ustindex].class,"Literal");
        UT[ustindex].index=j;
        ustindex++;
        break;
                }
                j++;
        }

        while(k<idindex) {
                if(strcmp(identifiers[k],tok)==0) {
        strcpy(UT[ustindex].name,tok);
        strcpy(UT[ustindex].class,"Identifier");
        UT[ustindex].index=k;
        ustindex++;
        break;
                }
                k++;
        }
}

void UST() {
        char buffer[80];
        char token[30];
        sp=fopen("input.txt","r");
        int i=0;
        while(fgets(buffer,80,sp)) {
      int j=0,t=0;
                int len=strlen(buffer);

      while(j<len-1) {
        if(buffer[j]==' ' || buffer[j]=='\t' || buffer[j]=='\n');
```

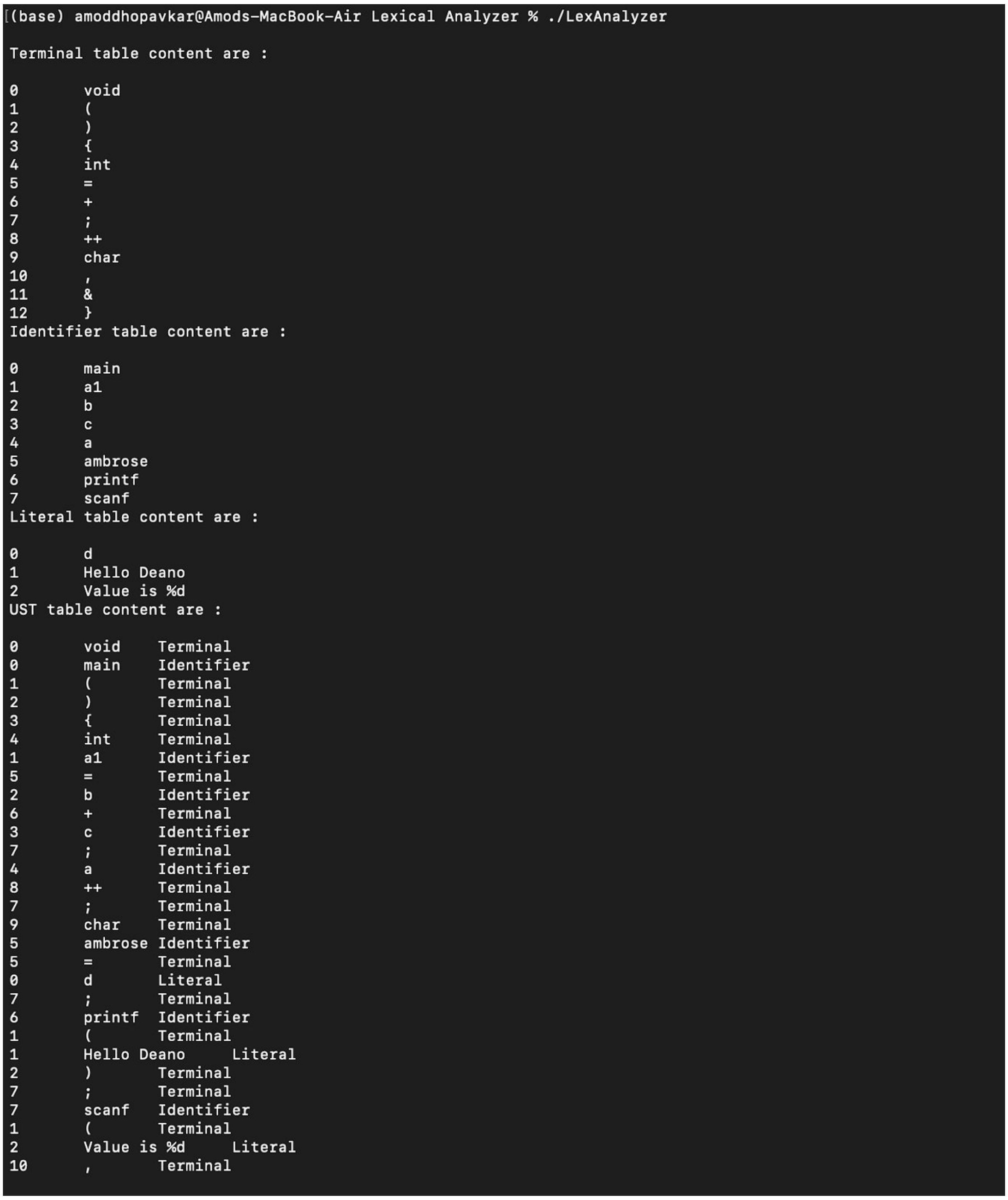
```
else if(isalpha(buffer[j])) {
    int flag=1;
    while(isalnum(buffer[j]) || buffer[j]=='_') {
        flag=0;
        token[t++]=buffer[j++];
    }
    if(flag==0)
        j--;
    token[t]='\0';
    t=0;
    check(token);
}

else if(isdigit(buffer[j])) {
    while(isdigit(buffer[j])) {
        token[t++]=buffer[j++];
    }
    token[t]='\0';
    t=0;
    check(token);
}

else {
    char demo[50];
    int di=0;
    if(!isdigit(buffer[j]) && (buffer[j]!=' ' || buffer[j]!='\n' || buffer[j]!='\t')) {
        if(buffer[j],buffer[j+1]=='+') {
            demo[0]=buffer[j++];
            demo[1]=buffer[j];
            demo[2]='\0';
            check(demo);
        }
        else {
            demo[0]=buffer[j];
            demo[1]='\0';
            check(demo);

        }

        if(buffer[j]=='\"') {
            j++;
            while(buffer[j]!='\"') {
                demo[di++]=buffer[j++];
            }
            demo[di]='\0';
```

```
                        check(demo);
                    }

                    if(buffer[j]=='\"') {
                        j++;
                        while(buffer[j]!='\"') {
                            demo[di++]=buffer[j++];
                        }
                        demo[di]='\0';
                        check(demo);
                    }
                }
            }
            j++;
                        }
            }
            fclose(sp);
}

int main() {
    construct();
    int i=5;
    printf("\nTerminal table content are :\n");
    int j;
    UST();
    for(i=0;i<trindex;i++) {
        printf("\n%d \t%s",i,trm[i]);
    }

    printf("\nIdentifier table content are :\n");
    for(i=0;i<idindex;i++) {
        printf("\n%d\t%s",i,identifiers[i]);
    }

    printf("\nLiteral table content are :\n");
    for(i=0;i<ltindex;i++) {
        printf("\n%d\t%s",i,literals[i]);
    }

    printf("\nUST table content are :\n");
    for(i=0;i<ustindex;i++) {
        printf("\n%d\t%s\t%s",UT[i].index,UT[i].name,UT[i].class);
    }
}
```

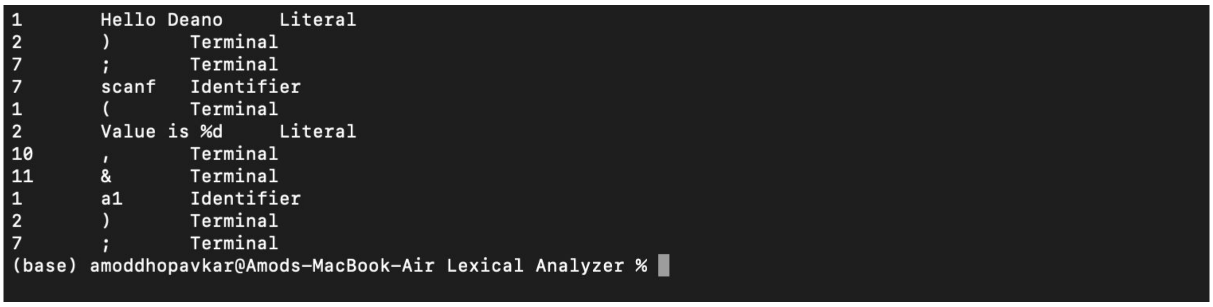
Input→

```
void main()
{
	int a1=b+c;
	a++;
	char ambrose='d';
	printf("Hello Deano");
	scanf("Value is %d",&a1);
}
```

Output→



```
[(base) amoddhopavkar@Amods-MacBook-Air Lexical Analyzer % ./LexAnalyzer

Terminal table content are :

0       void
1       (
2       )
3       {
4       int
5       =
6       +
7       ;
8       ++
9       char
10      ,
11      &
12      }
Identifier table content are :

0       main
1       a1
2       b
3       c
4       a
5       ambrose
6       printf
7       scanf
Literal table content are :

0       d
1       Hello Deano
2       Value is %d
UST table content are :

0       void    Terminal
0       main    Identifier
1       (       Terminal
2       )       Terminal
3       {       Terminal
4       int     Terminal
1       a1      Identifier
5       =       Terminal
2       b       Identifier
6       +       Terminal
3       c       Identifier
7       ;       Terminal
4       a       Identifier
8       ++      Terminal
7       ;       Terminal
9       char    Terminal
5       ambrose Identifier
5       =       Terminal
0       d       Literal
7       ;       Terminal
6       printf  Identifier
1       (       Terminal
1       Hello Deano     Literal
2       )       Terminal
7       ;       Terminal
7       scanf   Identifier
1       (       Terminal
2       Value is %d     Literal
10      ,       Terminal
```



```
1       Hello Deano     Literal
2       )       Terminal
7       ;       Terminal
7       scanf   Identifier
1       (       Terminal
2       Value is %d     Literal
10      ,       Terminal
11      &       Terminal
1       a1      Identifier
2       )       Terminal
7       ;       Terminal
(base) amoddhopavkar@Amods-MacBook-Air Lexical Analyzer %
```