NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL (MANGALORE)

COMPILER DESIGN PROJECT
REPORT 1 – LEXICAL ANALYSIS

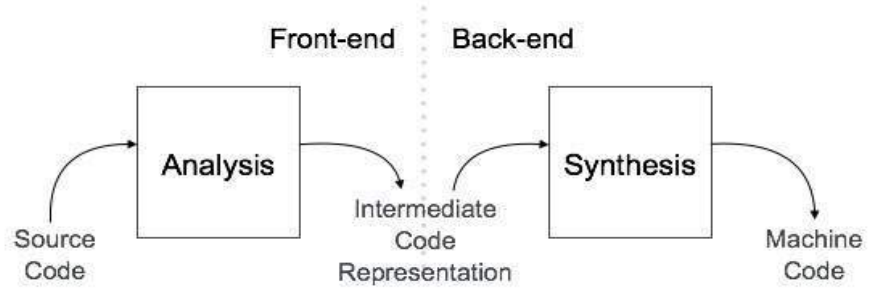
TEAM MEMBERS : -
KAPIL VASHIST (15CO123)
GAUTHAM M (15CO118)

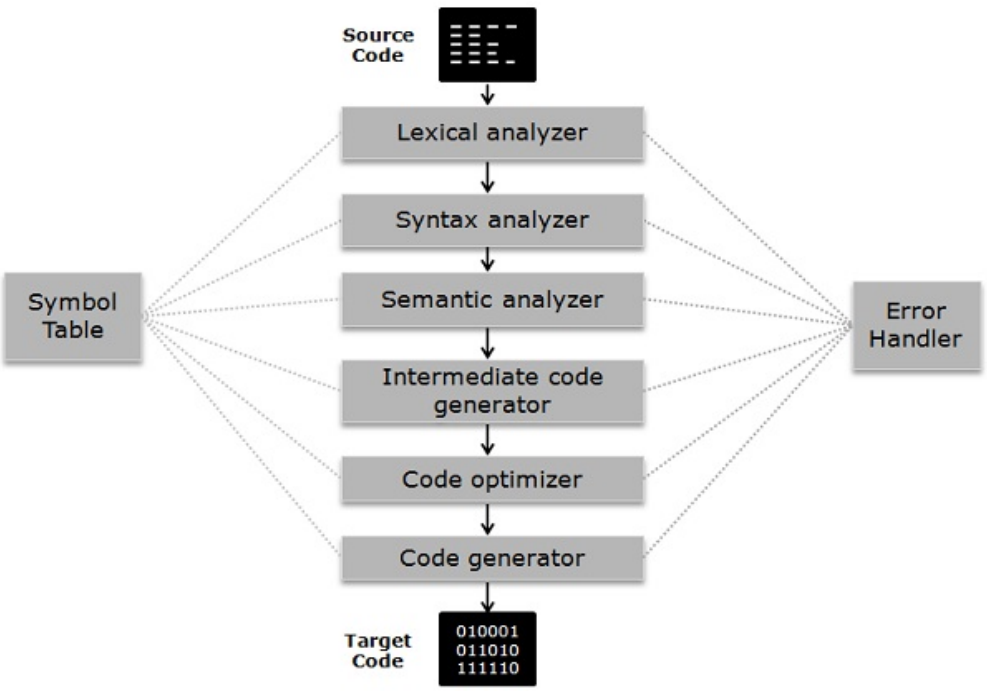
INTRODUCTION TO COMPILER

A Compiler is a language translator. It converts any high level language code into an equivalent and correct machine level language code.

A Compiler can be broadly divided into two phases :-
1. Analysis Phase
2. Synthesis Phase

ANALYSIS PHASE	SYNTHESIS PHASE
1. Lexical Analysis **2.** Syntactical Analysis **3.** Semantic Analysis **4.** Intermediate Code Generation	**1.** Code Optimization **2.** Code Generation

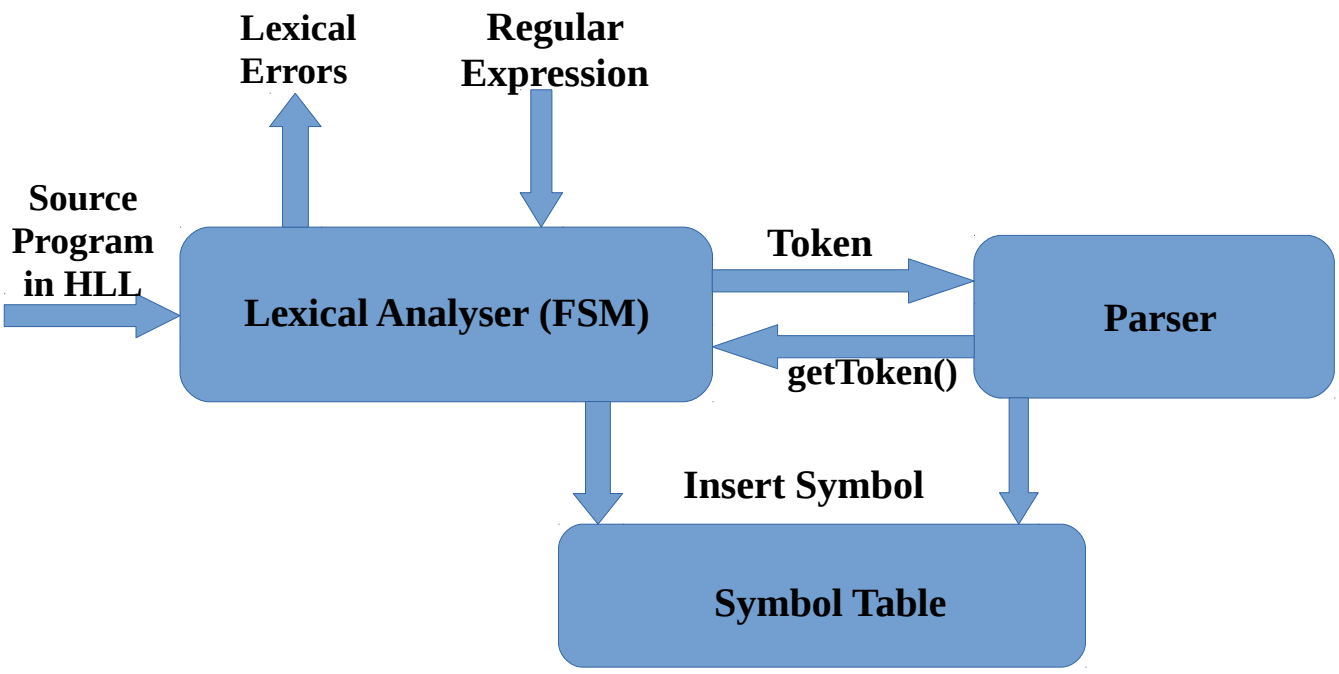




LEXICAL ANALYSIS : -

Lexical analysis is the first phase of a compiler. It takes the modified source code from language preprocessors that are written in the form of sentences. The lexical analyzer breaks these syntaxes into a series of tokens, by removing any whitespace or comments in the source code.

If the lexical analyzer finds a token invalid, it generates an error. The lexical analyzer works closely with the syntax analyzer. It reads character streams from the source code, checks for legal tokens, and passes the data to the syntax analyzer when it demands.



Lexemes are said to be a sequence of characters (alphanumeric) in a token. There are some predefined rules for every lexeme to be identified as a valid token. These rules are defined by grammar rules, by means of a pattern. A pattern explains what can be a token, and these patterns are defined by means of regular expressions.

In programming language, keywords, constants, identifiers, strings, numbers, operators and punctuations symbols can be considered as tokens.

For example, in C language, the variable declaration line

int value = 100;

contains the tokens:

int (keyword), value (identifier), = (operator), 100 (constant) and ; (symbol).

Longest Match Rule

When the lexical analyzer read the source-code, it scans the code letter by letter; and when it encounters a whitespace, operator symbol, or special symbols, it decides that a word is completed.

For example:

int intvalue;

While scanning both lexemes till ‘int’, the lexical analyzer cannot determine whether it is a keyword *int* or the initials of identifier int value.

The Longest Match Rule states that the lexeme scanned should be determined based on the longest match among all the tokens available.

The lexical analyzer also follows **Rule Priority** where a reserved word, e.g., a keyword, of a language is given priority over user input. That is, if the lexical analyzer finds a lexeme that matches with any existing reserved word, it should generate an error.

Source Code:

```
%{
#include<stdio.h>
#include<string.h>
int c=0;
int l=0;
int comment_stack[100];//for storing comment stack
int stacktop=-1;//top of comment stack
int commentflag=0;//check if thing is inside commnet
//hash function
unsigned long hash(unsigned char *str)
{
    unsigned long hash = 5381;
    int c;

    while (c = *str++)
        hash = ((hash << 5) + hash) + c; /* hash * 33 + c */

    return hash;
}

//symbol table
struct symbol{
int valid;
char name[100];
char type[100];
int lineno[100];
int linecount;
struct symbol *next;
} symboltable[65535];

//constants table
struct constants{
int valid;
char name[100];
char type[100];
int lineno[100];
int linecount;
struct constants *next;
} constantstable[65535];

//no of constants
int countconstants=0;
int countsymbol=0;
```

```
//function to add symbol to symbol table
void push_to_symbol_table(char ctemp[],unsigned long map,char type[])
{
	if(symboltable[map].valid==1&&strcmp(symboltable[map].name,ctemp)==0) //case when
the symbol is already in table first
	{

		symboltable[map].lineno[symboltable[map].linecount]=l;
		symboltable[map].linecount++;

	}
	else if(symboltable[map].valid==1&&strcmp(symboltable[map].name,ctemp)!=0)//case
when symbol not in table first
	{
		int found=0;
		struct symbol * pointer=&symboltable[map];
		while(pointer->next!=NULL)
		{
			if(strcmp(pointer->name,ctemp)==0)
				{
					found=1;
				}
				pointer=pointer->next;
		}
		if(found==0)
			////case when symbol not found
		{
			struct symbol * tempsymbol;
			strncpy(tempsymbol->name,yytext,yyleng);
			strcpy(tempsymbol->type,type);
			tempsymbol->linecount=0;
			tempsymbol->valid=1;
			tempsymbol->next=NULL;
			tempsymbol->lineno[tempsymbol->linecount]=l;
			tempsymbol->linecount++;
			pointer->next=tempsymbol;
		}
		if(found==1)
			//case when symbol found at some other position
		{
			pointer->lineno[pointer->linecount]=l;
			pointer->linecount++;

		}

	}
	else{
		  //hashtable map is free
		strncpy(symboltable[map].name,yytext,yyleng);
		strcpy(symboltable[map].type,type);
		symboltable[map].linecount=0;
		symboltable[map].valid=1;
		symboltable[map].next=NULL;
		symboltable[map].lineno[symboltable[map].linecount]=l;
		symboltable[map].linecount++;
		++countsymbol;
		}
}
```

```
//function to add constants to constant table
void push_to_constants_table(char ctemp[],unsigned long map,char type[])
                                        //case when the symbol is already in table first
{
        if(constantstable[map].valid==1&&strcmp(constantstable[map].name,ctemp)==0)
                        {
                                constantstable[map].lineno[constantstable[map].linecount]=l;
                                constantstable[map].linecount++;


                        }
                        else
if(constantstable[map].valid==1&&strcmp(constantstable[map].name,ctemp)!=0)          //case
when symbol not in table first
                        {
                                int found=0;
                                struct constants * pointer=&constantstable[map];
                                while(pointer->next!=NULL)
                                {
                                        if(strcmp(pointer->name,ctemp)==0)
                                                {
                                                        found=1;
                                                }
                                        pointer=pointer->next;
                                }
                                if(found==0)
                                                ////case when symbol not found
                                {
                                        struct constants * tempconstants;
                                        strncpy(tempconstants->name,yytext,yyleng);
                                        strcpy(tempconstants->type,type);
                                        tempconstants->linecount=0;
                                        tempconstants->valid=1;
                                        tempconstants->next=NULL;
                                        tempconstants->lineno[tempconstants->linecount]=l;
                                        tempconstants->linecount++;
                                        pointer->next=tempconstants;
                                }
                                if(found==1)
                                        //case when symbol found at some other position

                                {
                                        pointer->lineno[pointer->linecount]=l;
                                        pointer->linecount++;

                                }

                        }
                        else{
                                                                        //hashtable map is free
                                strncpy(constantstable[map].name,yytext,yyleng);
                                strcpy(constantstable[map].type,type);
                                constantstable[map].linecount=0;
                                constantstable[map].valid=1;
                                constantstable[map].next=NULL;
                                constantstable[map].lineno[constantstable[map].linecount]=l;
                                constantstable[map].linecount++;
                                ++countconstants;
                                }
}
```

```
%}
preprocessordirective #.*
single_line_comments \/\/.*
multi_line_comments_start \/\*
multi_line_comments_end \*\/
space [\ ]|\t
line \n
openparanth [{]
closeparanth [}]
strings ["].*["]
keyword void|for|do|while|if|else|return|auto|break|case|char|const|continue|default|double|enum|
extern|float|goto|if|int|long|register|return|short|signed|sizeof|static|struct|switch|typedef|union|
unsigned|volatile
intconst [0-9]+
floatconst [0-9]+[\.][0-9]+
charconst [a-z|A-Z]
functions [a-z][a-zA-Z0-9]*\(.*\)
specialsymbols \(|\)|\]|\[|,
delimitter ;
Operator [\+|-|=|/|\*|%]
notIdentifier [^a-zA-Z\t\n\ "\(|\)|\]|\[|,|"|\+|\-|=|/|\*|%|>|<|&|\||_|;][a-zA-Z0-9]*[a-zA-Z][a-zA-Z0-9]*
identifier [a-z][a-zA-Z0-9]*
error .
%%
{line} {
        ++l;     //increment line count
        }
{space} {}
{preprocessordirective} {
        if(commentflag==0){
        char ctemp[100];
        strncpy(ctemp,yytext,yyleng);
        unsigned long map=hash(ctemp);
        map=map%65535;
        push_to_symbol_table(ctemp,map,"pre processor directory");
        memset(&ctemp[0], 0, sizeof(ctemp));
        }
        else
        {
                printf("%s",yytext);
        }

}
{openparanth} {
if(commentflag==0){
        char ctemp[100];
        strncpy(ctemp,yytext,yyleng);
        unsigned long map=hash(ctemp);
        map=map%65535;
        push_to_symbol_table(ctemp,map,"openParanthesis");
        memset(&ctemp[0], 0, sizeof(ctemp));
        }
        else
        {
                printf("%s",yytext);
        }
}
```

```
{closeparanth} {
if(commentflag==0){
	char ctemp[100];
	strcpy(ctemp," ");
	strncpy(ctemp,yytext,yyleng);
	unsigned long map=hash(ctemp);
	map=map%65535;
	push_to_symbol_table(ctemp,map,"closeparanthesis");
	memset(&ctemp[0], 0, sizeof(ctemp));
	}
	else
	{
		printf("%s",yytext);
	}
}
{multi_line_comments_start} {
	stacktop++;
	comment_stack[stacktop]=1;
	commentflag=1;
	printf("comment start at %d :",l);
}
{multi_line_comments_end} {
	if(stacktop==-1){
		printf("comment error at %d \n", l);
	}
	else{
		stacktop--;
		if(stacktop==-1)
			{
				printf("comment end at %d \n", l);
				commentflag=0;
				printf("\n");
			}
	}
}
{single_line_comments} {
	if(commentflag==0){
	printf("comment :%s at %d\n",yytext,l);

	}
	else
	{
		printf("%s",yytext);
	}
		++l;
}
```

```
{intconst} {
if(commentflag==0){
	char ctemp[100];
	strcpy(ctemp," ");
	strncpy(ctemp,yytext,yyleng);
	unsigned long map=hash(ctemp);
	map=map%65535;
	push_to_constants_table(ctemp,map,"integer constant");
	memset(&ctemp[0], 0, sizeof(ctemp));
	}
	else
	{
	printf("%s",yytext);
	}
}
{floatconst} {
if(commentflag==0){
	char ctemp[100];
	strcpy(ctemp," ");
	strncpy(ctemp,yytext,yyleng);
	unsigned long map=hash(ctemp);
	map=map%65535;
	push_to_constants_table(ctemp,map,"float constant");
	memset(&ctemp[0], 0, sizeof(ctemp));
	}
	else
	{
		printf("%s",yytext);
	}
}
{charconst} {
if(commentflag==0){
	char ctemp[100];
	strcpy(ctemp," ");
	strncpy(ctemp,yytext,yyleng);
	unsigned long map=hash(ctemp);
	map=map%65535;
	push_to_constants_table(ctemp,map,"character constant");
	memset(&ctemp[0], 0, sizeof(ctemp));
	}
	else
	{
		printf("%s",yytext);
	}
}
```

```
{strings} {
if(commentflag==0){
	char ctemp[100];
	strcpy(ctemp," ");
	strncpy(ctemp,yytext,yyleng);
	unsigned long map=hash(ctemp);
	map=map%65535;
	push_to_constants_table(ctemp,map,"string constant");
	memset(&ctemp[0], 0, sizeof(ctemp));
	}
	else
	{
		printf("%s",yytext);
	}
}
{specialsymbols} {
if(commentflag==0){
	char ctemp[100];
	strcpy(ctemp," ");
	strncpy(ctemp,yytext,yyleng);
	unsigned long map=hash(ctemp);
	map=map%65535;
	push_to_symbol_table(ctemp,map,"special symbols");
	memset(&ctemp[0], 0, sizeof(ctemp));
	}
	else
	{
		printf("%s",yytext);
	}
}
{delimitter} {
}
{keyword} {
if(commentflag==0){
	char ctemp[100];
	strcpy(ctemp," ");
	strncpy(ctemp,yytext,yyleng);
	unsigned long map=hash(ctemp);
	map=map%65535;
	push_to_symbol_table(ctemp,map,"keywords");
	memset(&ctemp[0], 0, sizeof(ctemp));
	}
	else
	{
		printf("%s",yytext);
	}
}
```

```
{operator} {
if(commentflag==0){
	char ctemp[100];
	strcpy(ctemp," ");
	strncpy(ctemp,yytext,yyleng);
	unsigned long map=hash(ctemp);
	map=map%65535;
	push_to_symbol_table(ctemp,map,"operator");
	memset(&ctemp[0], 0, sizeof(ctemp));
	}
	else
	{
		printf("%s",yytext);
	}
}
{notIdentifier} {
	if(commentflag==0){
	printf("Error:Invalid Identifier%s\n ",yytext);
	}
	else{
	printf("%s",yytext);
	}

}
{identifier} {
if(commentflag==0){
	char ctemp[100];
	strcpy(ctemp," ");
	strncpy(ctemp,yytext,yyleng);
	unsigned long map=hash(ctemp);
	map=map%65535;
	push_to_symbol_table(ctemp,map,"identifier");
	memset(&ctemp[0], 0, sizeof(ctemp));
	}
	else
	{
		printf("%s",yytext);
	}
}
{error} {
	if(commentflag==0){
		printf("error at %s  %d\n",yytext,l);
	}
}
%%
int main()
{
	yyin=fopen("example4.c","r");
	yylex();
	if(stacktop!=-1)
	{
		printf("comment doesnt match\n");
	}
	printf("\n\t\t\t\t\tsymbols table\n");
	printf("%s \t\t %s \t\t %s \t\t %s \t\t %s \n","ID","name","type","linecount","linenumbers");
```

```
for(int i=0;i<65535;++i)
	{
			if(symboltable[i].valid==1)
			{
				struct symbol * pointer=&symboltable[i];
				while(pointer->next!=NULL)
					{
						printf("%d \t\t %s \t\t %s \t\t %d ",i,pointer->name,pointer->type,pointer->linecount);
						for(int j=0;j<pointer->linecount;++j)
						{
							printf("\t%d ",pointer->lineno[j]);
						}
							pointer=pointer->next;

					}
				printf("%d \t\t %s \t\t %s\t\t %d ",i,pointer->name,pointer->type,pointer->linecount);
						for(int j=0;j<pointer->linecount;++j)
						{
							printf("\t%d ",pointer->lineno[j]);
						}
							pointer=pointer->next;

				printf("\n");
			}
	}
	printf("\n\n\t\t\t\tconstant table\n");
	printf("%s \t\t %s \t\t %s \t\t %s \t\t %s \n","ID","name","type","linecount","linenumbers");
	for(int i=0;i<65535;++i)
	{
			if(constantstable[i].valid==1)
			{
				struct constants * pointer=&constantstable[i];
				while(pointer->next!=NULL)
					{
						printf("%d \t\t %s \t\t %s \t\t %d ",i,pointer->name,pointer->type,pointer->linecount);
						for(int j=0;j<pointer->linecount;++j)
						{
							printf("\t%d ",pointer->lineno[j]);
						}
						pointer=pointer->next;

					}
				printf("%d \t\t %s \t\t %s \t\t %d ",i,pointer->name,pointer->type,pointer->linecount);
```

```
for(int j=0;j<pointer->linecount;++j)
                                        {
                                                printf("\t%d ",pointer->lineno[j]);
                                        }
                                        pointer=pointer->next;
                                        printf("\n");
                                }
        }
        printf("count of lines is %d\n",l);
        return 0;
}
int yywrap()
{
        return(1);
}
```

Testing and output screen

Example 1 code:

```
#include<stdio.h>
int main()
{
        printf("Compiler Design");
        return 0;
}
```

Screenshot

```
gautham@gautham-Inspiron-5559:~/Desktop/CD$ ./a.out

                                        symbols table
ID              name            type            linecount               linenumbers
18294           printf                   identifier              1      3
33704           return                   keywords                1      4
35768           int             keywords                1       1
46543           (               special symbols                 2       1       3
46544           )               special symbols                 2       1       3
46626           {               openParanthesis                 1       2
46628           }               closeparanthesis                1       5
58812           #include<stdio.h>               pre processor directory                 1       0
64517           main            identifier              1       1


                                        constant table
ID              name            type            linecount               linenumbers
46551           0               integer constant                1       4
53152           "Compiler Design"               string constant                 1       3
count of lines is 6
gautham@gautham-Inspiron-5559:~/Desktop/CD$
```

Example 2 Code:

```
#include<stdio.h>
int main()
{
        int a=10;
        printf("lucky number is %d",a);
        return 0;
}
```

Screenshot

```
gautham@gautham-Inspiron-5559:~/Desktop/CD$ ./a.out

                                symbols table
ID              name            type            linecount               linenumbers
18294           printf                   identifier              1      4
33704           return                   keywords                1      5
35768           int             keywords                2       1      3
46543           (               special symbols                  2      1       4
46544           )               special symbols                  2      1       4
46547           ,               special symbols                  1      4
46564           =               operator                1       3
46626           {               openParanthesis                  1      2
46628           }               closeparanthesis                 1      6
58812           #include<stdio.h>               pre processor directory                 1       0
64517           main            identifier              1       1


                                constant table
ID              name            type            linecount               linenumbers
28824           "lucky number is %d"            string constant                 1       4
28959           10              integer constant                1       3
46551           0               integer constant                1       5
46600           a               character constant              2       3       4
count of lines is 7
gautham@gautham-Inspiron-5559:~/Desktop/CD$
```

Example 3 Code:

```
#include<stdio.h>
int main()
{
        //i am a student of NITK.
        /*Currently
        in 6th semester.
        aah!!!*/
        float inta;
        scanf("%d",&inta);
        return 0;
}
```

Screenshot

```
gautham@gautham-Inspiron-5559:~/Desktop/CD$ ./a.out example3.c
comment ://i am a student of NITK. at 3
comment start at 5 :Currentlyin6thsemesteraahcomment end at 7


                                symbols table
ID              name            type            linecount               linenumbers
811             inta            identifier              2       8       9
1881            scanf           identifier              1       9
33704           return                  keywords                1       10
35768           int             keywords                1       1
46541           &               operator                1       9
46543           (               special symbols                 2       1       9
46544           )               special symbols                 2       1       9
46547           ,               special symbols                 1       9
46626           {               openParanthesis                 1       2
46628           }               closeparanthesis                1       11
58812           #include<stdio.h>               pre processor directory                 1       0
61757           float           keywords                1       8
64517           main            identifier              1       1


                                constant table
ID              name            type            linecount               linenumbers
46551           0               integer constant                1       10
56131           "%d"            string constant                 1       9
count of lines is 12
gautham@gautham-Inspiron-5559:~/Desktop/CD$
```

Example 4 Code:

```c
#include<stdio.h>
int main()
{
        float abc=1.1.1;
        int a = 0x0g;

        printf("Compiler Design");
        return 0;
}
```

Screenshot

```
gautham@gautham-Inspiron-5559:~/Desktop/CD$ ./a.out example4.c
error at .  3
Error:Invalid Identifier0x0g

                                symbols table
ID              name            type            linecount               linenumbers
18294           printf                  identifier              1       6
26643           abc             identifier              1       3
33704           return                  keywords                1       7
35768           int             keywords                2       1       4
46543           (               special symbols                 2       1       6
46544           )               special symbols                 2       1       6
46564           =               operator                2       3       4
46626           {               openParanthesis                 1       2
46628           }               closeparanthesis                1       8
58812           #include<stdio.h>               pre processor directory         1       0
61757           float           keywords                1       3
64517           main            identifier              1       1


                                constant table
ID              name            type            linecount               linenumbers
38140           1.1             float constant                  1       3
46551           0               integer constant                1       7
46552           1               integer constant                1       3
46600           a               character constant              1       4
53152           "Compiler Design"               string constant                 1       6
count of lines is 9
gautham@gautham-Inspiron-5559:~/Desktop/CD$
```

Example 5 code

```
#include<stdio.h>
int main()
{
	char c[10]="ab\qcd";
	int a[100];
	for(int n=0;n<10;++n)
	{
		scanf("%d",&a[i]);
	}
	int ctr=0;
	while(ctr<10)
	{
		printf("%d",a[i]);
	++ctr;
	}
	printf("Compiler Design");
	return 0;
}
```

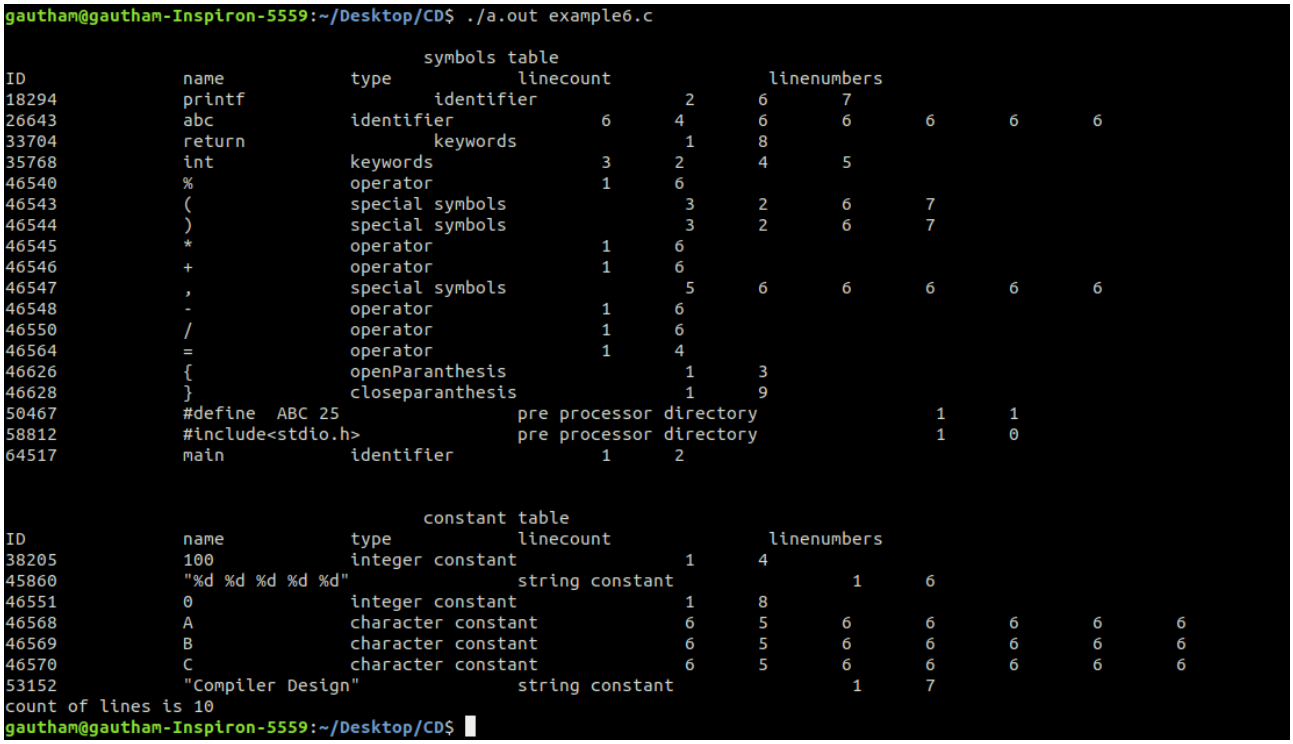
Screenshot

```
gautham@gautham-Inspiron-5559:~/Desktop/CD$ ./a.out

                                symbols table
ID              name            type            linecount               linenumbers
1881            scanf           identifier              1       7
18294           printf                  identifier              2       12      15
18770           while           keywords                1       10
29430           ctr             identifier              3       9       10      13
32532           for             keywords                1       5
33704           return                  keywords                1       16
35768           int             keywords                4       1       4       5       9
40185           char            keywords                1       3
46541           &               operator                1       7
46543           (               special symbols                 6       1       5       7       10      12      15
46544           )               special symbols                 6       1       5       7       10      12      15
46546           +               operator                4       5       5       13      13
46547           ,               special symbols                 2       7       12
46563           <               operator                2       5       10
46564           =               operator                3       3       5       9
46594           [               special symbols                 4       3       4       7       12
46596           ]               special symbols                 4       3       4       7       12
46626           {               openParanthesis                 3       2       6       11
46628           }               closeparanthesis                3       8       14      17
58812           #include<stdio.h>               pre processor directory                 1       0
64517           main            identifier              1       1


                                constant table
ID              name            type            linecount               linenumbers
28954           "ab\qcd"                string constant                 1       3
28959           10              integer constant                3       3       5       10
38205           100             integer constant                1       4
46551           0               integer constant                3       5       9       16
46600           a               character constant              3       4       7       12
46602           c               character constant              1       3
46608           i               character constant              2       7       12
46613           n               character constant              3       5       5       5
53152           "Compiler Design"               string constant                 1       15
56131           "%d"            string constant                 2       7       12
count of lines is 18
gautham@gautham-Inspiron-5559:~/Desktop/CD$
```

Example 6 Code:

```
#include<stdio.h>
#define  ABC 25
int main()
{
        int abc=100;
        int ABC;
        printf("%d %d %d %d %d",abc-ABC,abc+ABC,abc*ABC,abc/ABC,abc%ABC);
        printf("Compiler Design");
        return 0;
}
```

Screenshot



```
gautham@gautham-Inspiron-5559:~/Desktop/CD$ ./a.out example6.c

                                symbols table
ID              name            type            linecount               linenumbers
18294           printf                  identifier              2       6       7
26643           abc             identifier              6       4       6       6       6       6       6
33704           return                  keywords                1       8
35768           int             keywords                3       2       4       5
46540           %               operator                1       6
46543           (               special symbols                 3       2       6       7
46544           )               special symbols                 3       2       6       7
46545           *               operator                1       6
46546           +               operator                1       6
46547           ,               special symbols                 5       6       6       6       6       6
46548           -               operator                1       6
46550           /               operator                1       6
46564           =               operator                1       4
46626           {               openParanthesis                 1       3
46628           }               closeparanthesis                1       9
50467           #define  ABC 25                 pre processor directory                 1       1
58812           #include<stdio.h>               pre processor directory                 1       0
64517           main            identifier              1       2


                                constant table
ID              name            type            linecount               linenumbers
38205           100             integer constant                1       4
45860           "%d %d %d %d %d"                string constant                 1       6
46551           0               integer constant                1       8
46568           A               character constant              6       5       6       6       6       6       6
46569           B               character constant              6       5       6       6       6       6       6
46570           C               character constant              6       5       6       6       6       6       6
53152           "Compiler Design"               string constant                 1       7
count of lines is 10
gautham@gautham-Inspiron-5559:~/Desktop/CD$
```

Example7 Code:

```
#include<stdio.h>
int add(int a,int b)
{
        return a+b;
}
int main()
{
        int a=10,b=23;
        printf("lucky number is %d",a);
        int a8b=add(a,b);
        return 0;
}
```

Screenshot

```
gautham@gautham-Inspiron-5559:~/Desktop/CD$ ./a.out

                                    symbols table
ID              name            type            linecount               linenumbers
18294           printf                   identifier              1      8
25256           a8b             identifier              1      9
26710           add             identifier              2      1       9
33704           return                   keywords                2      3       10
35768           int             keywords                6      1       1       1       5       7       9
46543           (               special symbols                 4      1       5       8       9
46544           )               special symbols                 4      1       5       8       9
46546           +               operator                1      3
46547           ,               special symbols                 4      1       7       8       9
46564           =               operator                3      7       7       9
46626           {               openParanthesis                 2      2       6
46628           }               closeparanthesis                2      4       11
58812           #include<stdio.h>               pre processor directory                 1      0
64517           main            identifier              1      5


                                    constant table
ID              name            type            linecount               linenumbers
28824           "lucky number is %d"            string constant                 1      8
28959           10              integer constant                1      7
28995           23              integer constant                1      7
46551           0               integer constant                1      10
46600           a               character constant              5      1       3       7       8       9
46601           b               character constant              4      1       3       7       9
count of lines is 12
gautham@gautham-Inspiron-5559:~/Desktop/CD$
```

Example 8 Code:

```
#include<stdio.h>
int main()
{
//tester
/*boom
b
a*/
*/
/*
anda
*/


        funct ();
        int a1bc=0;
        int a[100];
        a1bc=a1bc+10;
        int ;
        int;
        printf("\n abc %d",a1bc);


        return 0;
}
```

Screenshot

```
gautham@gautham-Inspiron-5559:~/Desktop/CD$ ./a.out
comment ://tester at 3
comment start at 5 :boombacomment end at 7

comment error at 8
comment start at 9 :andacomment end at 11


                                symbols table
ID              name            type            linecount               linenumbers
18294           printf                  identifier              1       20
33704           return                  keywords                1       23
35768           int             keywords                5       1       15      16      18      19
39504           a1bc            identifier              4       15      17      17      20
46543           (               special symbols                 3       1       14      20
46544           )               special symbols                 3       1       14      20
46546           +               operator                1       17
46547           ,               special symbols                 1       20
46564           =               operator                2       15      17
46594           [               special symbols                 1       16
46596           ]               special symbols                 1       16
46626           {               openParanthesis                 1       2
46628           }               closeparanthesis                1       24
56492           funct           identifier              1       14
58812           #include<stdio.h>               pre processor directory                 1       0
64517           main            identifier              1       1


                                constant table
ID              name            type            linecount               linenumbers
28959           10              integer constant                1       17
38205           100             integer constant                1       16
46551           0               integer constant                2       15      23
46600           a               character constant              1       16
53516           "\n abc %d"             string constant                 1       20
count of lines is 25
gautham@gautham-Inspiron-5559:~/Desktop/CD$
```