Shreya Thigale
TE-IT
33168

PAGE NO.
DATE / /

ASSIGNMENT-8

TITLE : Lexical Analysis

PROBLEM STATEMENT:
Write a program to implement a lexical analyzer for subset of 'C' language.

OBJECTIVE:
1. To understand the basic principles in compilation
2. To study lexical analysis phase of compiler.

THEORY:

Compiler takes input as source program & produces output as an equivalent sequence of machine instructions. This process consists of two-step processing of source program:

1. ANALYSIS STEP
It consists of three substeps
1. lexical Analysis - Determine lexical constituents in source program
2. Syntax Analysis - Determine structure of source string
3. Semantic Analysis - Determine meaning of source string

2. SYNTHESIS STEP
It deals with memory allocation & code generation. The actions in analysis phase are uniquely defined for given language. But

Shreya Thigale
TE-IX
33168

PAGE NO.
DATE / /

LEXICAL ANALYSIS:

The action of scanning the source program into proper syntactic classes is known as lexical analysis.

Task of lexical analysis -
1. To scan the program into basic elements.
2. To build Uniform Symbol Table (UST)
3. To build the symbol & literal table
4. To remove white spaces and comments
5. To detect errors such as invalid identifiers or constant.

DATA STRUCTURES:

1. Source program - Original source program, which is scanned by compiler as string of characters.
2. Terminal table - A permanent database that has entry for each terminal symbols such as arithmetic operators, keywords, punctuation characters such as '(', ';' etc
3. Literal table - This table is created during lexical analysis so as to describe all literals in program
4. Identifier table - Created during lexical analysis and describes all identifiers in the program.
5. Uniform Symbol table - Created during lexical analysis to represent the program as a string of

Shreya Thigale
TE-IX
33168

PAGE No.
DATE / /

tokens, rather than individual characters.

6. Buffer - One buffer or two buffer schemes to load source program part by part to reduce disk I/O.

ALGORITHM:

1. Initialize line no to 1.
2. Read the source program line by line
3. For each line separate the tokens such as
   i) identifier / function name / keywords
   ii) Integer constant
   iii) All types of operators
   iv) Remove comments
   v) Remove all white spaces
4. Assign line no and increment line number
5. Repeat steps 2-4 till end of file

SAMPLE INPUT:

A program in C language

```
main()
{
  int i,no,sum,max;
  sum=0; max=-32767;
  for (i=0;i<10; i++)
  {
    scanf("%d",&no);
    sum = sum+no;
    if (max<no) max=no;
  }
```

Shreya Thigale
TE-IX
33168

PAGE No.
DATE / /

```
printf("sum = %d  max = %d\n", sum, max);
getch();
}
```

Terminal Table

(
)
{
}
+
-
=
&
,
;
<
>
"
++

Keyword Table

for
if
int
float

Identifier Table

identifier	Attribute
Main	Function name
i	
no	
sum	
max	
scanf	Function name
printf	Function name
getch	Function name

Literal Table

Literal	Attribute
0	Numeric constant
-32767	Numeric constant
10	Numik constant
"%d"	String constant
"sum = %d max = %d\n"	String constant

Shreya Thigale
TE-IX
33168

PAGE No.
DATE / /

Uniform Symbol Table

	TYPE	INDEX
main	IDN	0
(	TRM	0
)	TRM	1
{	TRM	2
int	KEY	0
i	IDN	1
,	TRM	8
no	IDN	2
,	TRM	8
sum	IDN	3
,	TRM	8
max	IDN	4
;	TRM	9
sum	IDN	3
=	TRM	6
0	LIT	0
;	TRM	9
max	IDN	4
=	TRM	6
-32767	LIT	1
;	TRM	9
for	KEY	0
(	TRM	0
i	IDN	1
=	TRM	6
0	LIT	0
;	TRM	9
:	;	;
:		

Shreya Thigale
TE-IX
33168

PAGE No.
DATE / /

CONCLUSION:

Thus we have successfully implemented this experiment of lexical analyzer for subset of 'C' language. We have also studied various phases of lexical analysis and basic principles of compiler.