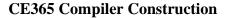


#### **Charotar University of Science and Technology**

# Devang Patel Institute of Advance Technology and Research

## **Department of Computer Engineering**





### Practical 5

### AIM:

Implementation of a Lexical Analyzer for C Language Compiler

### **OBJECTIVE:**

To design and implement a lexical analyser to perform 1st, 2nd, 3rd, and 5th task as per the list given in practical 2.

### **TESTCASES:**

```
/* salary calculation*/
void main()

{
long int bs , da , hra , gs;
//take basic salary as input
scanf("%ld",&bs);
//calculate allowances

// user defined data type
struct student
{
int id;
float cgpa;
}
void main()
```

```
da=bs*.40;
hra=bs*.20;
                                        student s;
gs=bs+da+hra;
                                        s.id = 10:
// display salary slip
                                        s.cgpa = 8.7;
printf("\n\nbs : %ld",bs);
printf("\nda : %ld",da);
printf("\nhra: %ld",hra);
printf("\ngs : %ld",gs);
//function prototype
void add (int, int);
void main( )
int a, b;
a = 10;
b = 20;
// function call
add (a, b);
void add (int x, int y)
return x + y;
```

**22DCE119** 

```
CODE:
%{
#include <stdio.h>
#include <string.h>
#include <ctype.h>
void printToken(const char* type, const char* value) {
  printf("%s: %s\n", type, value);
%}
KEYWORDS
"int"|"char"|"return"|"if"|"else"|"for"|"while"|"do"|"switch"|"case"|"break"|"continue"|"void"|"float"|"double"
IDENTIFIER [a-zA-Z][a-zA-Z0-9]*
CONSTANT [0-9]+
STRING
            \'[^']\'|\"[^\"]*\"
OPERATOR
              \+|\-|\*|\/|=|==|!=|<|>|<=|>=|&&|\|\
PUNCTUATION [\(\)\{\}\[\],;]
COMMENT
             \\.\\\([^]\\+[^\\])\+\\
%%
{KEYWORDS}
                    { printToken("Keyword", yytext); }
                  { printToken("Identifier", yytext); }
{IDENTIFIER}
                   { printToken("Constant", yytext); }
{CONSTANT}
                { printToken("String", yytext); }
{STRING}
{OPERATOR}
                   { printToken("Operator", yytext); }
{PUNCTUATION} { printToken("Punctuation", yytext); }
                   { /* Ignore comments */ }
{COMMENT}
            { /* Ignore white spaces */ }
\lceil t \rceil +
          { printToken("Lexical Error", yytext); } // Detect invalid tokens
%%
int main(int argc, char *argv[]) {
  if (argc < 2) {
    printf("Usage: %s <C source file>\n", argv[0]);
    return 1;
  FILE *file = fopen(argv[1], "r");
  if (!file) {
    printf("Error: Cannot open file %s\n", argv[1]);
    return 1;
  }
  yyin = file; // Set input file for Lex
            // Process file
  yylex();
  fclose(file); // Close file
```

```
return 0;
}
int yywrap() {
  return 1;
}
```

## **OUTPUT:**

```
Keyword: void
Identifier: main
Punctuation: (
Punctuation:
Punctuation: {
Identifier: long
Keyword: int
Identifier: bs
Punctuation: ,
Identifier: da
Punctuation: ,
Identifier: hra
Punctuation: ,
Identifier: gs
Punctuation: ;
Identifier: scanf
Punctuation: (
String: "%ld"
Punctuation: ,
Lexical Error: &
Identifier: bs
Punctuation: )
Punctuation: ;
Identifier: da
Operator: =
Identifier: bs
Operator: *
Lexical Error: .
Constant: 40
Punctuation: ;
Identifier: hra
Operator: =
Identifier: bs
Operator: *
Lexical Error: .
Constant: 20
Punctuation: ;
Identifier: gs
```

```
Operator: =
Identifier: bs
Operator: +
Identifier: da
Operator: +
Identifier: hra
Punctuation: ;
Identifier: printf
Punctuation: (
String: "\n\nbs : %ld"
Punctuation: ,
Identifier: bs
Punctuation: )
Punctuation: ;
Identifier: printf
Punctuation: (
String: "\nda : %ld"
Punctuation: ,
Identifier: da
Punctuation: )
Punctuation: ;
Identifier: printf
Punctuation: (
String: "\nhra : %ld"
Punctuation: ,
Identifier: hra
Punctuation: )
Punctuation: ;
Identifier: printf
Punctuation: (
String: "\ngs : %ld"
Punctuation: ,
Identifier: gs
Punctuation: )
Punctuation: ;
Punctuation: }
```