**Department of Computer Science and Engineering**

**National Institute of Technology, Hamirpur**



CSD - 319

Compiler Design Lab

Akash Rana

185034

# Experiment 1:

Develop a lexical analyzer to recognize a few patterns in c such as Identifiers, constants, comments, operators and ignore redundant spaces, tabs and new lines, comments etc.

## Code:

#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"};

int i, flag = 0;

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

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

flag = 1;

break;

}

}

return flag;

}

int main()

{

char ch, buffer[15], operators[] = "+-\*/%=";

FILE \*fp;

int i, j = 0;

fp = fopen("example\_1.txt", "r");

if (fp == NULL)

{

printf("error while opening the file\n");

exit(0);

}

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

{

for (i = 0; i < 6; ++i)

{

if (ch == operators[i])

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

}

if (isalnum(ch))

{

buffer[j++] = ch;

}

else if ((ch == ' ' || ch == '\n') && (j != 0))

{

buffer[j] = '\0';

j = 0;

if (isKeyword(buffer) == 1)

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

else

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

}

}

fclose(fp);

return 0;

}

## 

## 

## Input:

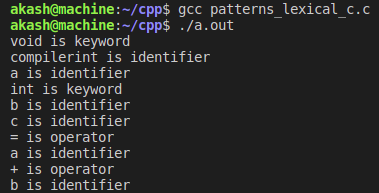
void compiler( int a, int b)

{

c = a + b;

}

## Output:



# Experiment 2:

Implement the lexical analyzer using Lex, flex or any other lexical analyzer generating tools.

## Code:

//Implementation of Lexical Analyzer using Lex tool

%{

int COMMENT=0;

%}

identifier [a-zA-Z][a-zA-Z0-9]\*

%%

#.\* {printf("\n%s is a preprocessor directive",yytext);}

int |

float |

char |

double |

while |

for |

struct |

typedef |

do |

if |

break |

continue |

void |

switch |

return |

else |

goto {printf("\n\t%s is a keyword",yytext);}

"/\*" {COMMENT=1;}{printf("\n\t %s is a COMMENT",yytext);}

{identifier}\( {if(!COMMENT)printf("\nFUNCTION \n\t%s",yytext);}

\{ {if(!COMMENT)printf("\n BLOCK BEGINS");}

\} {if(!COMMENT)printf("BLOCK ENDS ");}

{identifier}(\[[0-9]\*\])? {if(!COMMENT) printf("\n %s IDENTIFIER",yytext);}

\".\*\" {if(!COMMENT)printf("\n\t %s is a STRING",yytext);}

[0-9]+ {if(!COMMENT) printf("\n %s is a NUMBER ",yytext);}

\)(\:)? {if(!COMMENT)printf("\n\t");ECHO;printf("\n");}

\( ECHO;

= {if(!COMMENT)printf("\n\t %s is an ASSIGNMENT OPERATOR",yytext);}

\<= |

\>= |

\< |

== |

\> {if(!COMMENT) printf("\n\t%s is a RELATIONAL OPERATOR",yytext);}

%%

int main(int argc, char \*\*argv)

{

FILE \*file;

file=fopen("var.c","r");

if(!file)

{

printf("could not open the file");

exit(0);

}

yyin=file;

yylex();

printf("\n");

return(0);

}

int yywrap()

{

return(1);

}

## 

## Input:

#include <stdio.h>

#include <conio.h>

void main()

{

int a, b, c;

a = 1;

b = 2;

c = a + b;

printf("Sum:%d", c);

}

## Output:

## 