**Name:** D. Ashwanth Ram

**Reg No:** 3122 21 5001 013

—------------------------------------------------------------------------------------------------------------------

**Assignment-02: Implementation of lexical analyser using LEX tool**

---------------------------------------------------------------------------------------------------------------------

Aim:

**Lexer.l :-**

%{

#include <stdio.h>

#include <string.h>

int yylex(void);

void yyerror(const char \*s);

FILE \*yyin;

struct SymbolTableEntry {

char identifier[50];

char type[10];

int bytes;

char value[100];

int address;

};

struct SymbolTableEntry symbolTable[100];

int symbolCount = 0;

int currentAddress = 1000;

char lastIdentifier[50] = "";

char lastValue[100] = "";

void insertIntoSymbolTable(char \*identifier, char \*type, int bytes, char \*value, int address) {

for (int i = 0; i < symbolCount; i++) {

if (strcmp(symbolTable[i].identifier, identifier) == 0) {

if (strcmp(symbolTable[i].type, "variable") == 0) {

strcpy(symbolTable[i].value, value);

}

return;

}

}

strcpy(symbolTable[symbolCount].identifier, identifier);

strcpy(symbolTable[symbolCount].type, type);

symbolTable[symbolCount].bytes = bytes;

strcpy(symbolTable[symbolCount].value, value);

symbolTable[symbolCount].address = address;

symbolCount++;

}

void printSymbolTable() {

printf("\n----------------------------------------------------------\n");

printf("| %-20s | %-10s | %-5s | %-10s |\n", "Identifier", "Type", "Bytes", "Address");

printf("----------------------------------------------------------\n");

for (int i = 0; i < symbolCount; i++) {

printf("| %-20s | %-10s | %-5d | %-10d |\n",

symbolTable[i].identifier,

symbolTable[i].type,

symbolTable[i].bytes,

symbolTable[i].address);

}

printf("----------------------------------------------------------\n");

}

void setAddressForLastEntry() {

if (symbolCount > 0) {

symbolTable[symbolCount - 1].address = currentAddress;

currentAddress += 4;

}

}

%}

%option noyywrap

%%

"#include"<[a-zA-Z0-9]+\.h> {

printf("%s - header file\n", yytext);

}

"int"|"float"|"if"|"else"|"return"|"char"|"void"|"while"|"for"|"switch"|"case"|"break"|"continue"|"default"|"struct"|"union"|"typedef"|"enum"|"static"|"extern"|"sizeof"|"goto"|"register"|"auto"|"long"|"short"|"double"|"unsigned"|"signed" {

printf("%s - keyword\n", yytext);

}

[a-zA-Z\_][a-zA-Z0-9\_]\*\(\) {

if (strcmp(yytext, "printf()") != 0) {

printf("%s - function call\n", yytext);

}

}

[a-zA-Z\_][a-zA-Z0-9\_]\* {

if (strcmp(yytext, "printf") != 0) {

strcpy(lastIdentifier, yytext);

setAddressForLastEntry();

insertIntoSymbolTable(yytext, "variable", 4, "", currentAddress - 4);

printf("%s - identifier\n", yytext);

}

}

"=" {

printf("%s - assignment\n", yytext);

}

[0-9]+\.[0-9]+ {

printf("%s - float value\n", yytext);

strcpy(lastValue, yytext);

}

[0-9]+ {

printf("%s - integer value\n", yytext);

strcpy(lastValue, yytext);

}

\"[^\"]\*\" {

printf("%s - string value\n", yytext);

strcpy(lastValue, yytext);

}

"+"|"-"|"\*"|"/"|"%="|"+="|"-="|"\*="|"/="|"=="|"!="|"&&"|"||"|"<="|">="|"<"|">" {

printf("%s - operator\n", yytext);

}

"//".\* { /\* Ignore single-line comments \*/ }

"/\*"([^\*]\*\\*+([^/\*][^\*]\*\\*+)\*)?"\*/" {

if(!strstr(yytext, "\*/")) {

yyerror("Error: Unterminated comment");

}

}

[ \t\n]+ { /\* Ignore whitespace characters \*/ }

. {

printf("%s - special character\n", yytext);

}

%%

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

if (argc > 1) {

yyin = fopen(argv[1], "r");

if (!yyin) {

printf("Could not open file: %s\n", argv[1]);

return 1;

}

} else {

yyin = stdin;

}

yylex();

if (strlen(lastIdentifier) > 0 && strlen(lastValue) > 0) {

insertIntoSymbolTable(lastIdentifier, "variable", 4, lastValue, currentAddress - 4);

}

printSymbolTable();

return 0;

}

void yyerror(const char \*s) {

fprintf(stderr, "%s\n", s);

}

**Sample Input :-**

**#**include<stdio.h>

int main() {

int a = 10;

float b = 20.34;

char name[100] = "karthikeyan";

if (a < b)

printf("%d\n", a);

else if (a > b)

printf("%d\n", b);

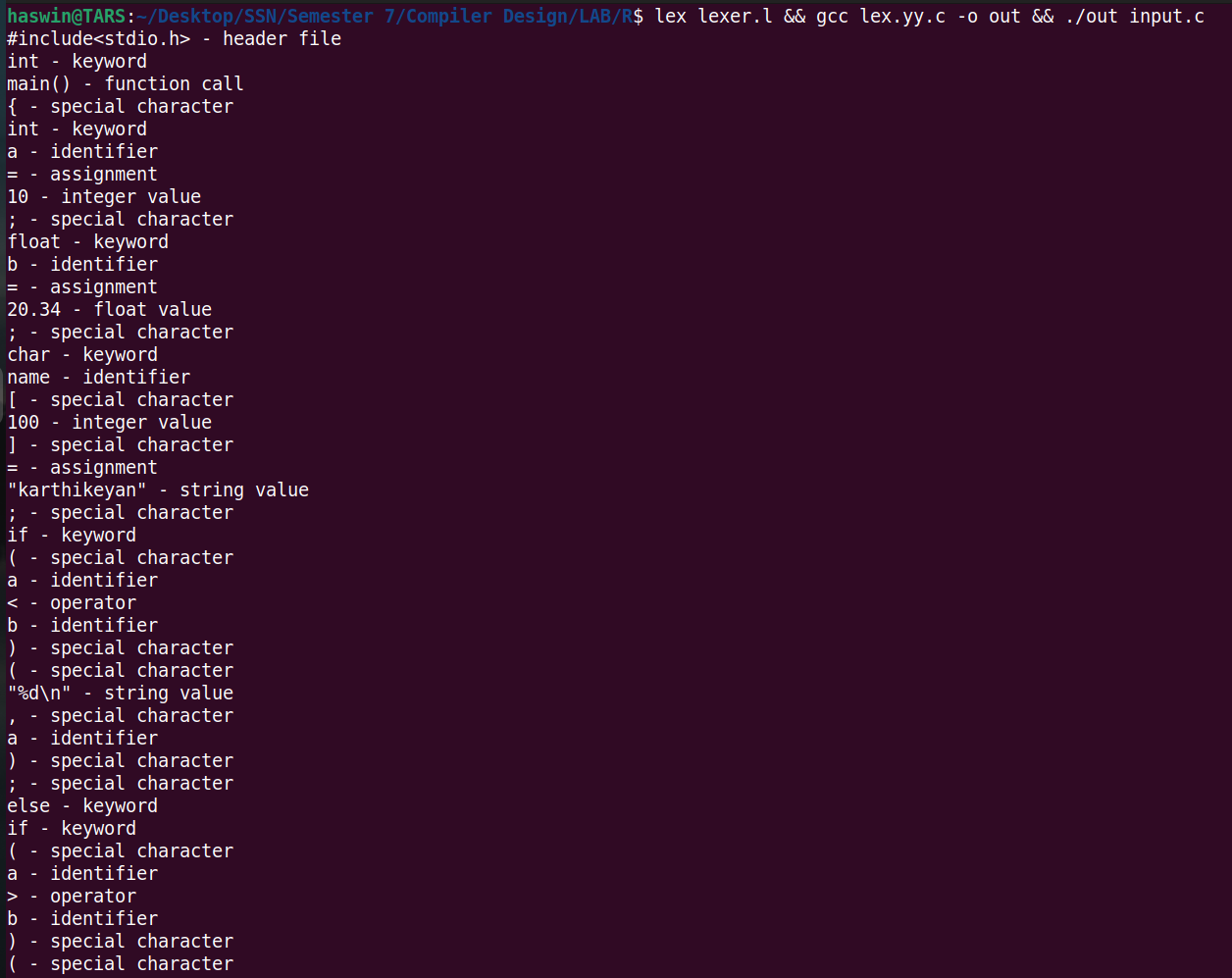
else

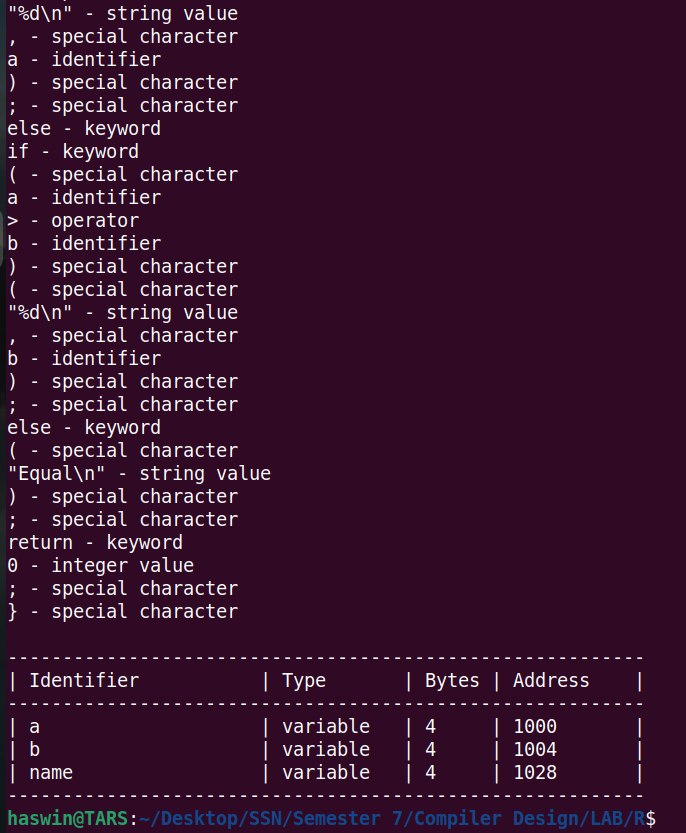
printf("Equal\n");

return 0;

}

**Output :-**

****

****

**Learning Outcomes :**

• Improved Lexical Analysis.

• Implemented Symbol Table Management

• Effective Value Assignment Handling