Mujtaba Shahid Faizi

BSCS-5A

#131818

Lab 4 of Compiler Construction

**TASK 1:**

**Code:**

%%

(("if"|("then")|("begin")|("end")|("procedure")|("else")|("int")|("char")|("fopen")|("stdin")|("goto")|("line")|("main")|("function"))) {printf("A keyword: "); ECHO; printf("\n\n");}

([+-/\*<>=]) {printf("An Operator: "); ECHO; printf("\n\n");}

((("<<")|(">>")|("^"))) {printf("A bitwise operator: "); ECHO; printf("\n\n");}

([\n\t" "]) {}

([(){};]) {printf("A punctuator: "); ECHO; printf("\n\n");}

(([0-9]+)(.[0-9]+)?) {printf("A number: "); ECHO; printf("\n\n");}

([a-zA-Z]+[0-9a-zA-Z]\*) {printf("An id: "); ECHO; printf("\n\n");}

. {printf("Unrecognized character: "); ECHO; printf("\n\n");}

%%

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

{

if(argc>1)

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

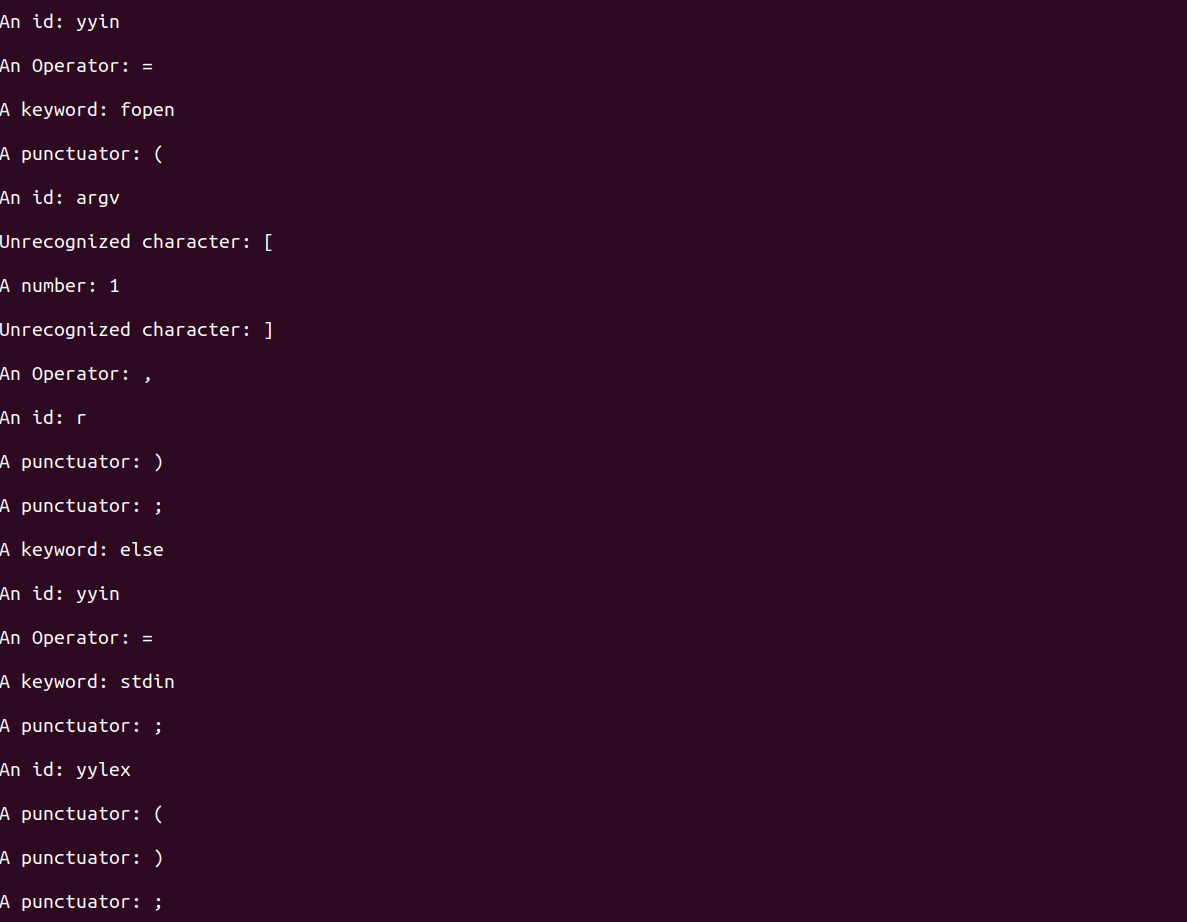
else

yyin=stdin;

yylex();

}

**Screenshot:**



# **TASK 2**

We can add definitions of our patterns, and then use them in the rules section. e.g.

operator [+-/\*><%=] to replace and reduce the redundancy with:

{operator} {printf("Operator:"). Thus, increasing the reusability of code.

e.g.

%{

#include<stdlib.h>

%}

operator [+-/\*><%=]

%%

{operator} {printf("An Operator: "); ECHO; printf("\n\n");}

%%