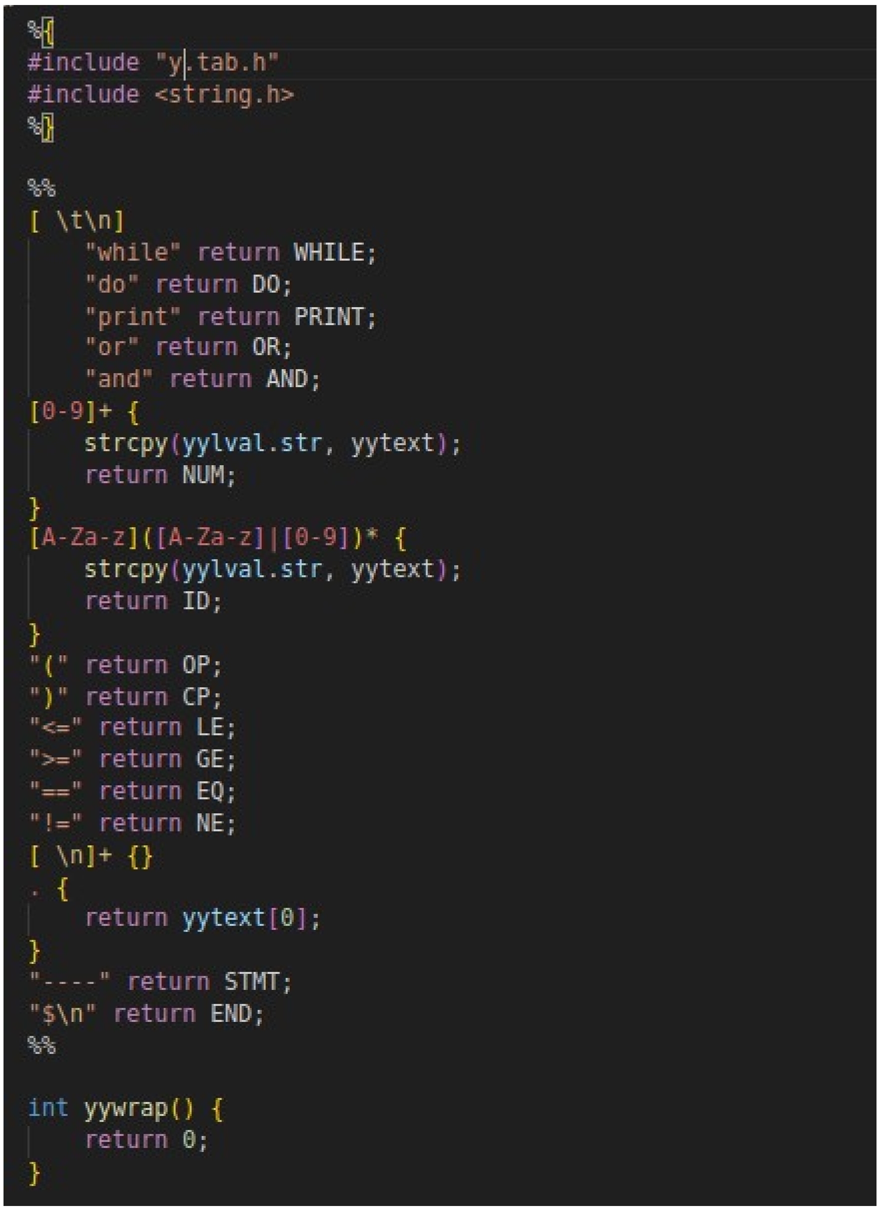
Nithin S
221IT085

IT250 Lab Assignment 8

Write LEX and YACC program to generate Intermediate code Generation for the following

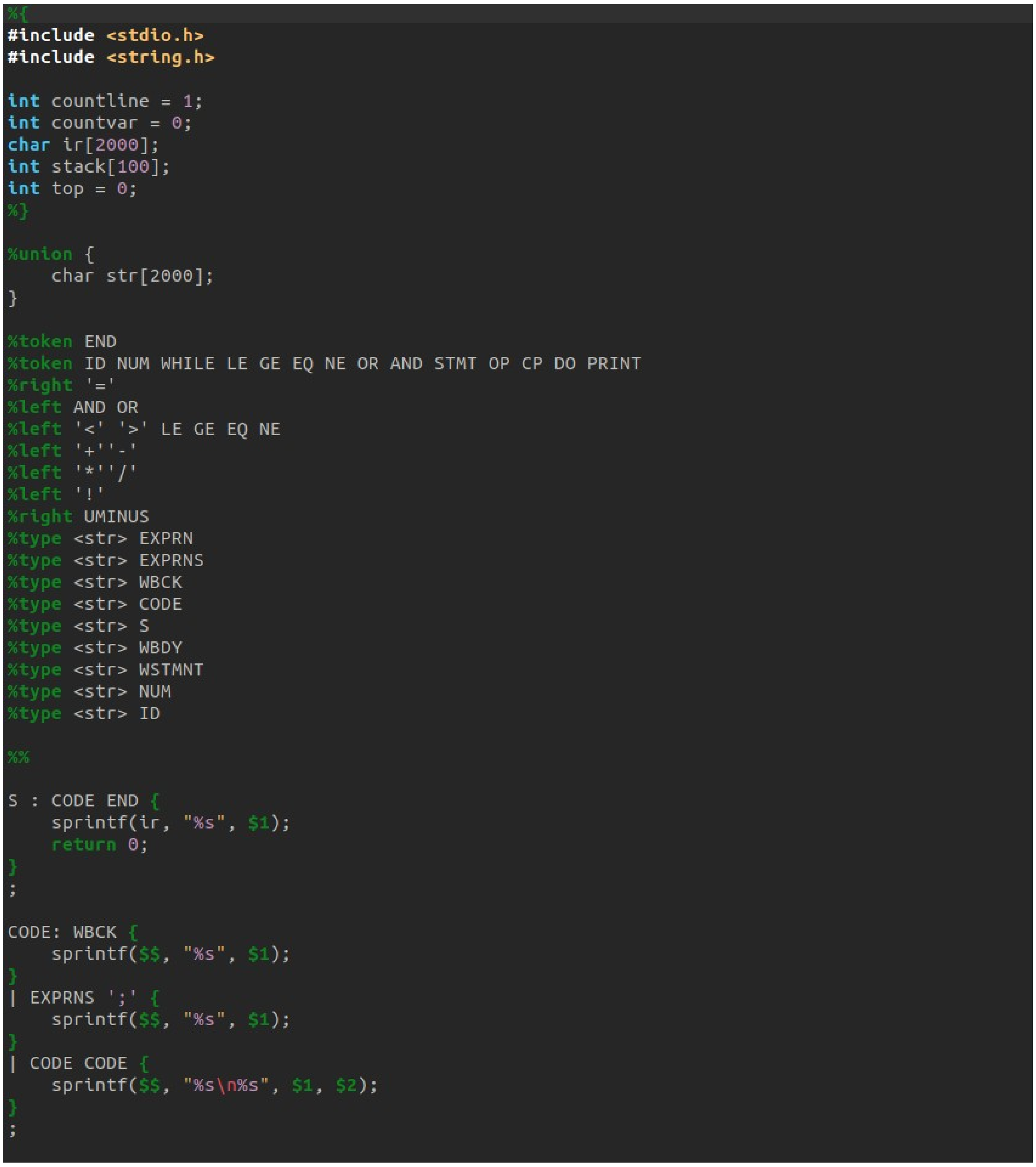
CODE

Lex Code

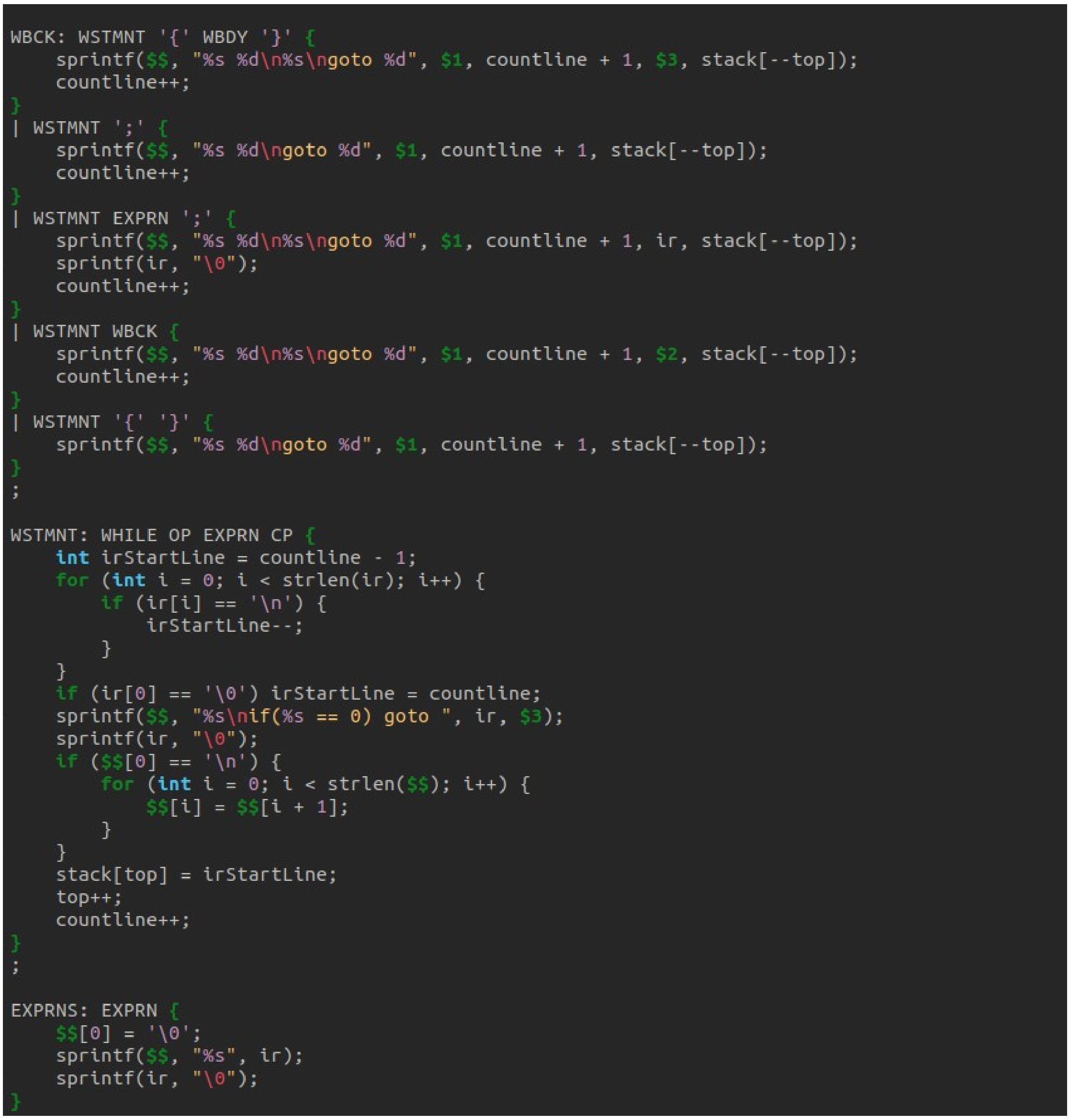


```
%{
#include "y.tab.h"
#include <string.h>
%}

%%
[ \t\n]
    "while" return WHILE;
    "do" return DO;
    "print" return PRINT;
    "or" return OR;
    "and" return AND;
[0-9]+ {
    strcpy(yylval.str, yytext);
    return NUM;
}
[A-Za-z]([A-Za-z]|[0-9])* {
    strcpy(yylval.str, yytext);
    return ID;
}
"(" return OP;
")" return CP;
"<=" return LE;
">=" return GE;
"==" return EQ;
"!=" return NE;
[ \n]+ {}
. {
    return yytext[0];
}
"----" return STMT;
"$\n" return END;
%%

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

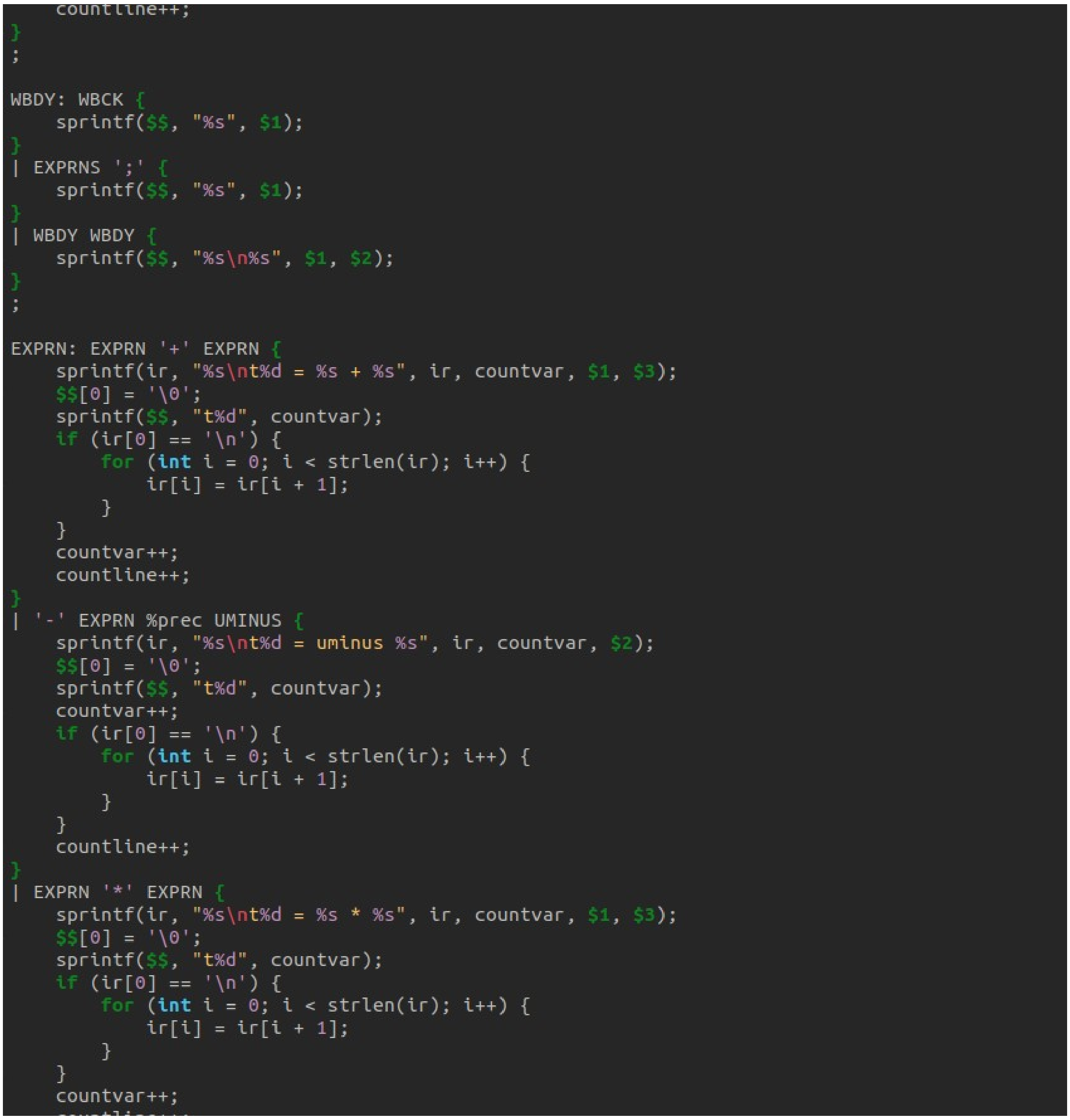
Yacc Code



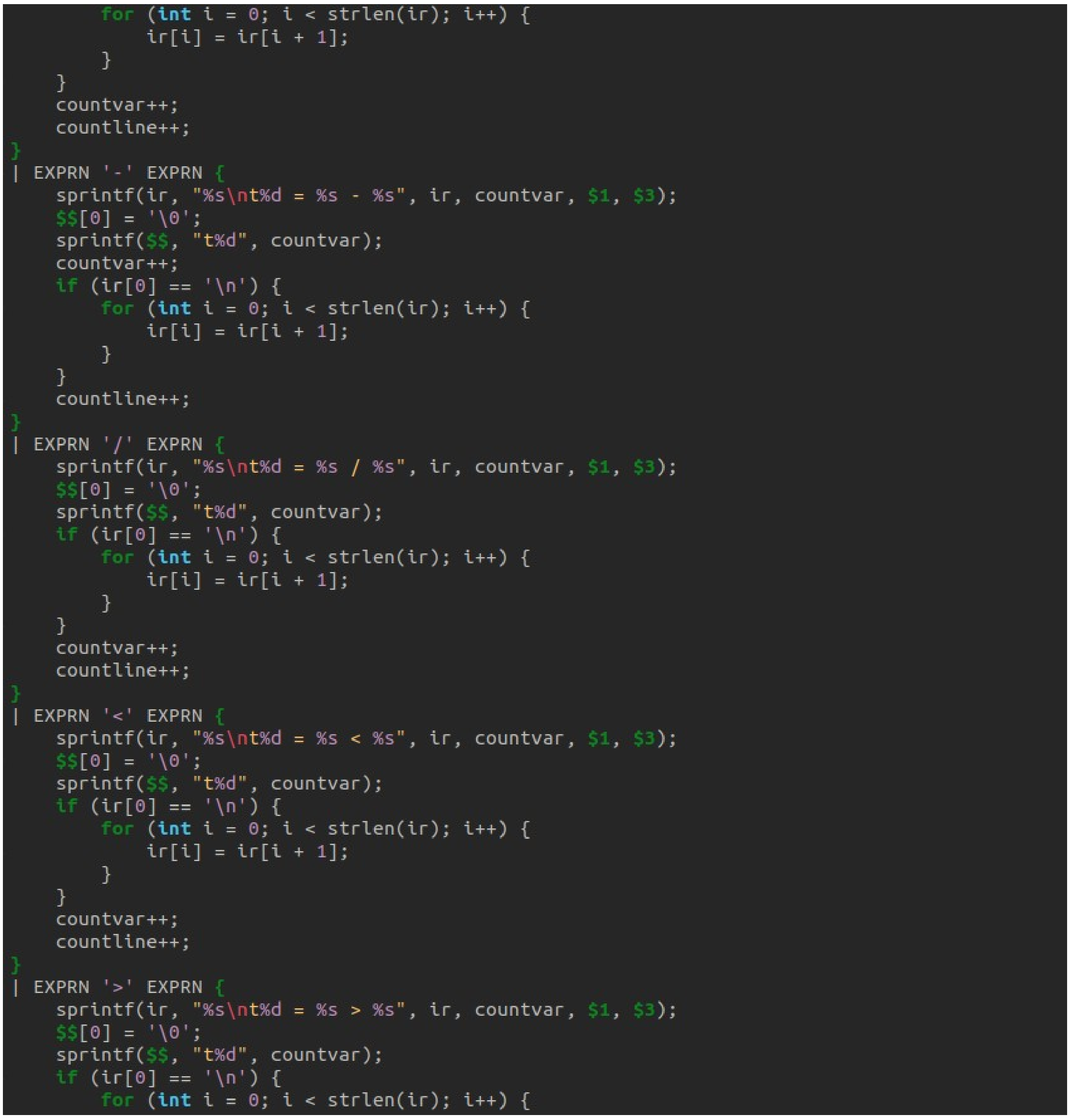
```
%{
#include <stdio.h>
#include <string.h>

int countline = 1;
int countvar = 0;
char ir[2000];
int stack[100];
int top = 0;
%}

%union {
    char str[2000];
}

%token END
%token ID NUM WHILE LE GE EQ NE OR AND STMT OP CP DO PRINT
%right '='
%left AND OR
%left '<' '>' LE GE EQ NE
%left '+''-'
%left '*''/'
%left '!'
%right UMINUS
%type <str> EXPRN
%type <str> EXPRNS
%type <str> WBCK
%type <str> CODE
%type <str> S
%type <str> WBDY
%type <str> WSTMNT
%type <str> NUM
%type <str> ID

%%

S : CODE END {
    sprintf(ir, "%s", $1);
    return 0;
}
;

CODE: WBCK {
    sprintf($$, "%s", $1);
}
| EXPRNS ';' {
    sprintf($$, "%s", $1);
}
| CODE CODE {
    sprintf($$, "%s\n%s", $1, $2);
}
;
```



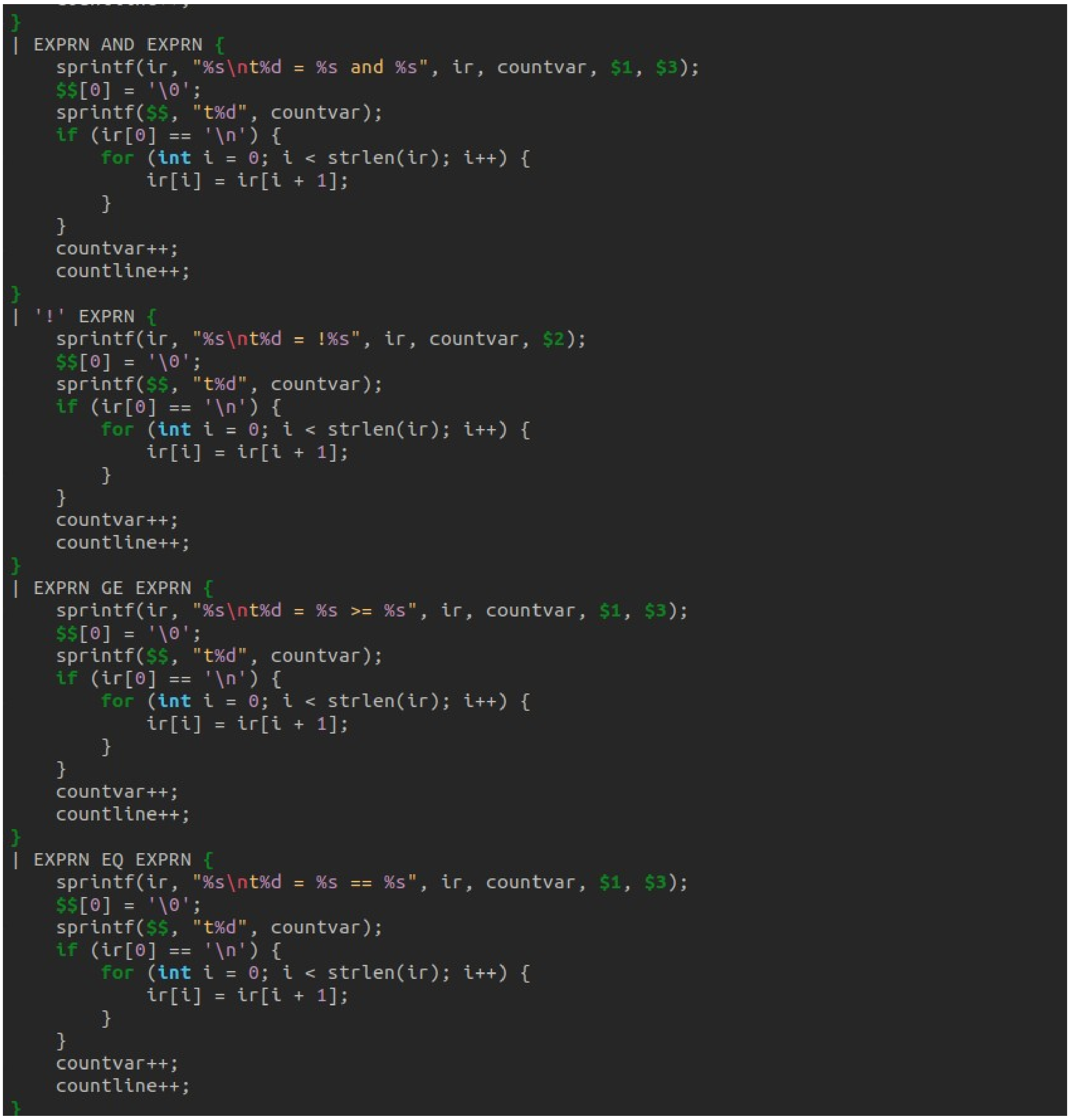
```
WBCK: WSTMNT '{' WBDY '}' {
    sprintf($$, "%s %d\n%s\ngoto %d", $1, countline + 1, $3, stack[--top]);
    countline++;
}
| WSTMNT ';' {
    sprintf($$, "%s %d\ngoto %d", $1, countline + 1, stack[--top]);
    countline++;
}
| WSTMNT EXPRN ';' {
    sprintf($$, "%s %d\n%s\ngoto %d", $1, countline + 1, ir, stack[--top]);
    sprintf(ir, "\0");
    countline++;
}
| WSTMNT WBCK {
    sprintf($$, "%s %d\n%s\ngoto %d", $1, countline + 1, $2, stack[--top]);
    countline++;
}
| WSTMNT '{' '}' {
    sprintf($$, "%s %d\ngoto %d", $1, countline + 1, stack[--top]);
}
;

WSTMNT: WHILE OP EXPRN CP {
    int irStartLine = countline - 1;
    for (int i = 0; i < strlen(ir); i++) {
        if (ir[i] == '\n') {
            irStartLine--;
        }
    }
    if (ir[0] == '\0') irStartLine = countline;
    sprintf($$, "%s\nif(%s == 0) goto ", ir, $3);
    sprintf(ir, "\0");
    if ($$[0] == '\n') {
        for (int i = 0; i < strlen($$); i++) {
            $$[i] = $$[i + 1];
        }
    }
    stack[top] = irStartLine;
    top++;
    countline++;
}
;

EXPRNS: EXPRN {
    $$[0] = '\0';
    sprintf($$, "%s", ir);
    sprintf(ir, "\0");
}
```



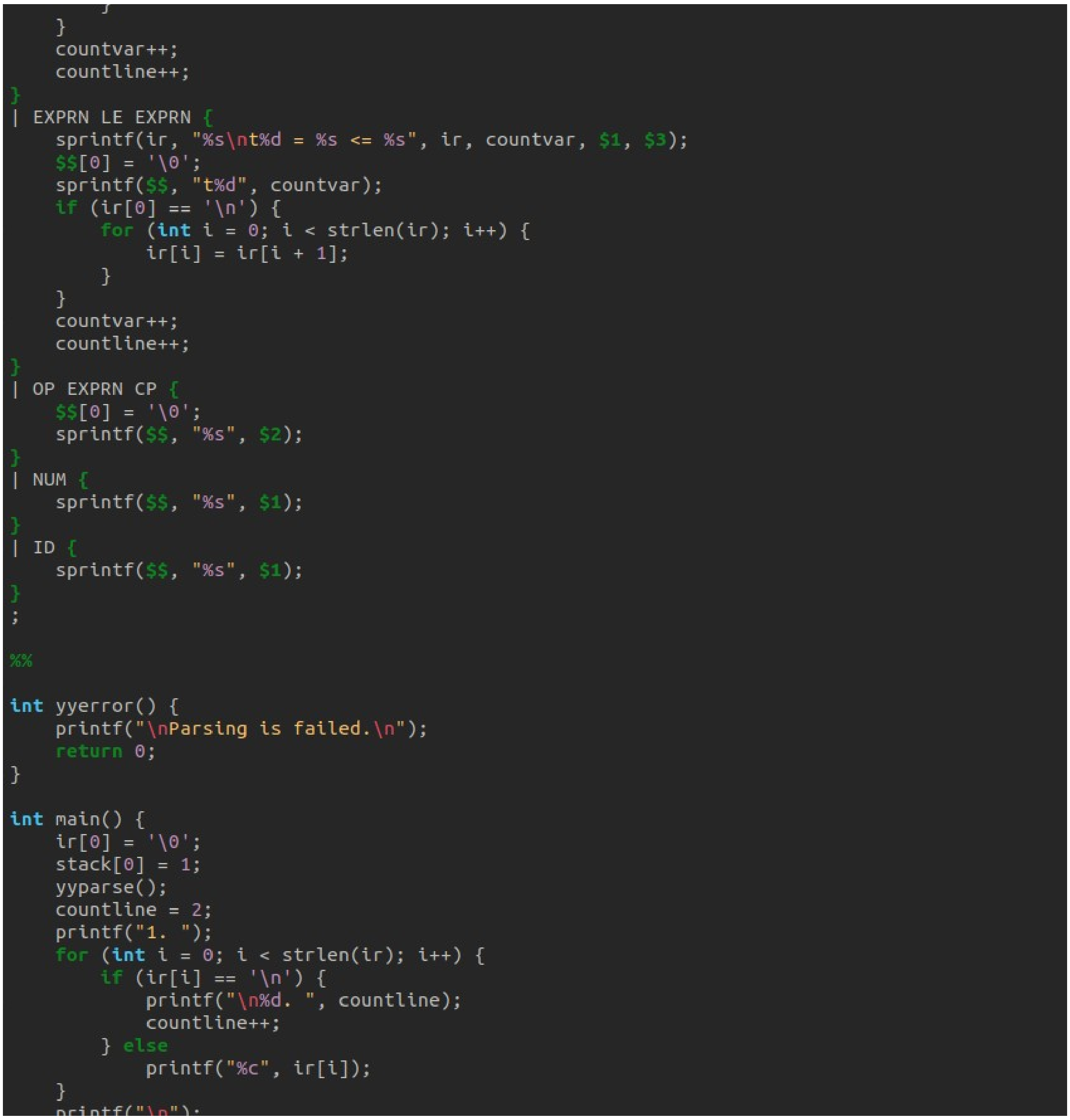
```
    countline++;
}
;

WBDY: WBCK {
    sprintf($$, "%s", $1);
}
| EXPRNS ';' {
    sprintf($$, "%s", $1);
}
| WBDY WBDY {
    sprintf($$, "%s\n%s", $1, $2);
}
;

EXPRN: EXPRN '+' EXPRN {
    sprintf(ir, "%s\nt%d = %s + %s", ir, countvar, $1, $3);
    $$[0] = '\0';
    sprintf($$, "t%d", countvar);
    if (ir[0] == '\n') {
        for (int i = 0; i < strlen(ir); i++) {
            ir[i] = ir[i + 1];
        }
    }
    countvar++;
    countline++;
}
| '-' EXPRN %prec UMINUS {
    sprintf(ir, "%s\nt%d = uminus %s", ir, countvar, $2);
    $$[0] = '\0';
    sprintf($$, "t%d", countvar);
    countvar++;
    if (ir[0] == '\n') {
        for (int i = 0; i < strlen(ir); i++) {
            ir[i] = ir[i + 1];
        }
    }
    countline++;
}
| EXPRN '*' EXPRN {
    sprintf(ir, "%s\nt%d = %s * %s", ir, countvar, $1, $3);
    $$[0] = '\0';
    sprintf($$, "t%d", countvar);
    if (ir[0] == '\n') {
        for (int i = 0; i < strlen(ir); i++) {
            ir[i] = ir[i + 1];
        }
    }
    countvar++;
```



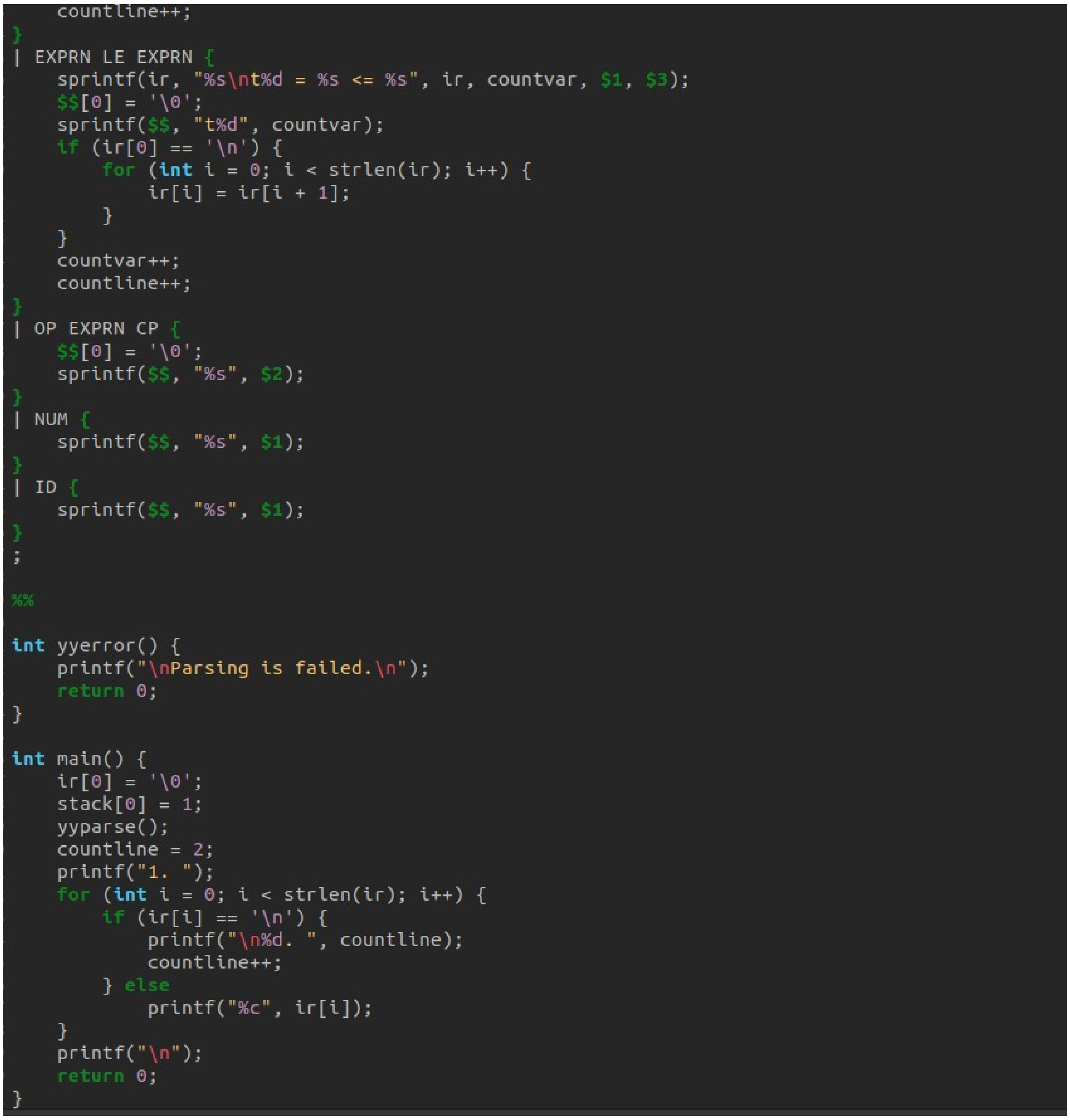
```
        for (int i = 0; i < strlen(ir); i++) {
            ir[i] = ir[i + 1];
        }
    }
    countvar++;
    countline++;
}
| EXPRN '-' EXPRN {
    sprintf(ir, "%s\nt%d = %s - %s", ir, countvar, $1, $3);
    $$[0] = '\0';
    sprintf($$, "t%d", countvar);
    countvar++;
    if (ir[0] == '\n') {
        for (int i = 0; i < strlen(ir); i++) {
            ir[i] = ir[i + 1];
        }
    }
    countline++;
}
| EXPRN '/' EXPRN {
    sprintf(ir, "%s\nt%d = %s / %s", ir, countvar, $1, $3);
    $$[0] = '\0';
    sprintf($$, "t%d", countvar);
    if (ir[0] == '\n') {
        for (int i = 0; i < strlen(ir); i++) {
            ir[i] = ir[i + 1];
        }
    }
    countvar++;
    countline++;
}
| EXPRN '<' EXPRN {
    sprintf(ir, "%s\nt%d = %s < %s", ir, countvar, $1, $3);
    $$[0] = '\0';
    sprintf($$, "t%d", countvar);
    if (ir[0] == '\n') {
        for (int i = 0; i < strlen(ir); i++) {
            ir[i] = ir[i + 1];
        }
    }
    countvar++;
    countline++;
}
| EXPRN '>' EXPRN {
    sprintf(ir, "%s\nt%d = %s > %s", ir, countvar, $1, $3);
    $$[0] = '\0';
    sprintf($$, "t%d", countvar);
    if (ir[0] == '\n') {
        for (int i = 0; i < strlen(ir); i++) {
```



```
}
| EXPRN AND EXPRN {
    sprintf(ir, "%s\nt%d = %s and %s", ir, countvar, $1, $3);
    $$[0] = '\0';
    sprintf($$, "t%d", countvar);
    if (ir[0] == '\n') {
        for (int i = 0; i < strlen(ir); i++) {
            ir[i] = ir[i + 1];
        }
    }
    countvar++;
    countline++;
}
| '!' EXPRN {
    sprintf(ir, "%s\nt%d = !%s", ir, countvar, $2);
    $$[0] = '\0';
    sprintf($$, "t%d", countvar);
    if (ir[0] == '\n') {
        for (int i = 0; i < strlen(ir); i++) {
            ir[i] = ir[i + 1];
        }
    }
    countvar++;
    countline++;
}
| EXPRN GE EXPRN {
    sprintf(ir, "%s\nt%d = %s >= %s", ir, countvar, $1, $3);
    $$[0] = '\0';
    sprintf($$, "t%d", countvar);
    if (ir[0] == '\n') {
        for (int i = 0; i < strlen(ir); i++) {
            ir[i] = ir[i + 1];
        }
    }
    countvar++;
    countline++;
}
| EXPRN EQ EXPRN {
    sprintf(ir, "%s\nt%d = %s == %s", ir, countvar, $1, $3);
    $$[0] = '\0';
    sprintf($$, "t%d", countvar);
    if (ir[0] == '\n') {
        for (int i = 0; i < strlen(ir); i++) {
            ir[i] = ir[i + 1];
        }
    }
    countvar++;
    countline++;
}
```

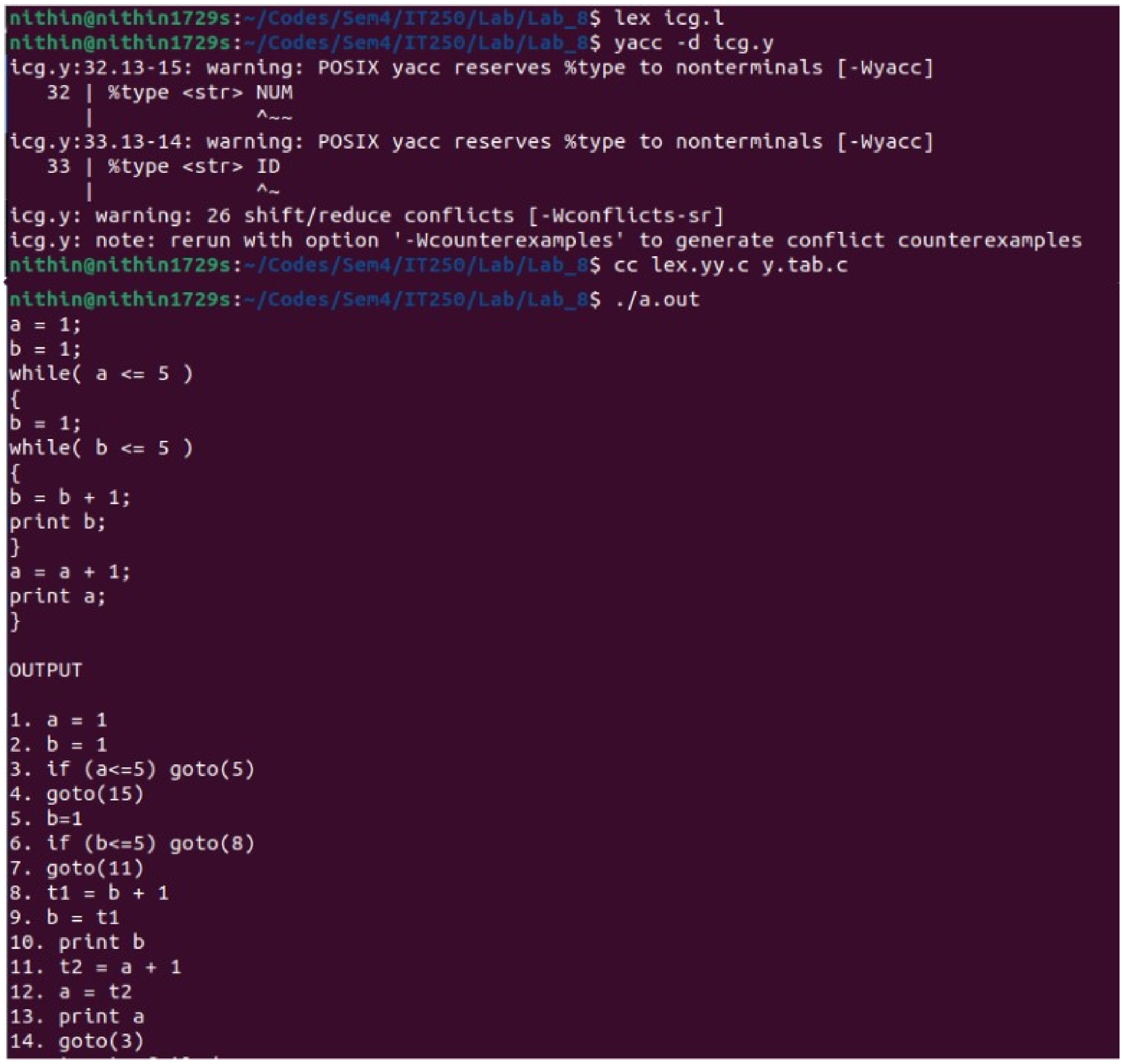


```
        }
    }
    countvar++;
    countline++;
}
| EXPRN LE EXPRN {
    sprintf(ir, "%s\nt%d = %s <= %s", ir, countvar, $1, $3);
    $$[0] = '\0';
    sprintf($$, "t%d", countvar);
    if (ir[0] == '\n') {
        for (int i = 0; i < strlen(ir); i++) {
            ir[i] = ir[i + 1];
        }
    }
    countvar++;
    countline++;
}
| OP EXPRN CP {
    $$[0] = '\0';
    sprintf($$, "%s", $2);
}
| NUM {
    sprintf($$, "%s", $1);
}
| ID {
    sprintf($$, "%s", $1);
}
;

%%

int yyerror() {
    printf("\nParsing is failed.\n");
    return 0;
}

int main() {
    ir[0] = '\0';
    stack[0] = 1;
    yyparse();
    countline = 2;
    printf("1. ");
    for (int i = 0; i < strlen(ir); i++) {
        if (ir[i] == '\n') {
            printf("\n%d. ", countline);
            countline++;
        } else
            printf("%c", ir[i]);
    }
    printf("\n");
```

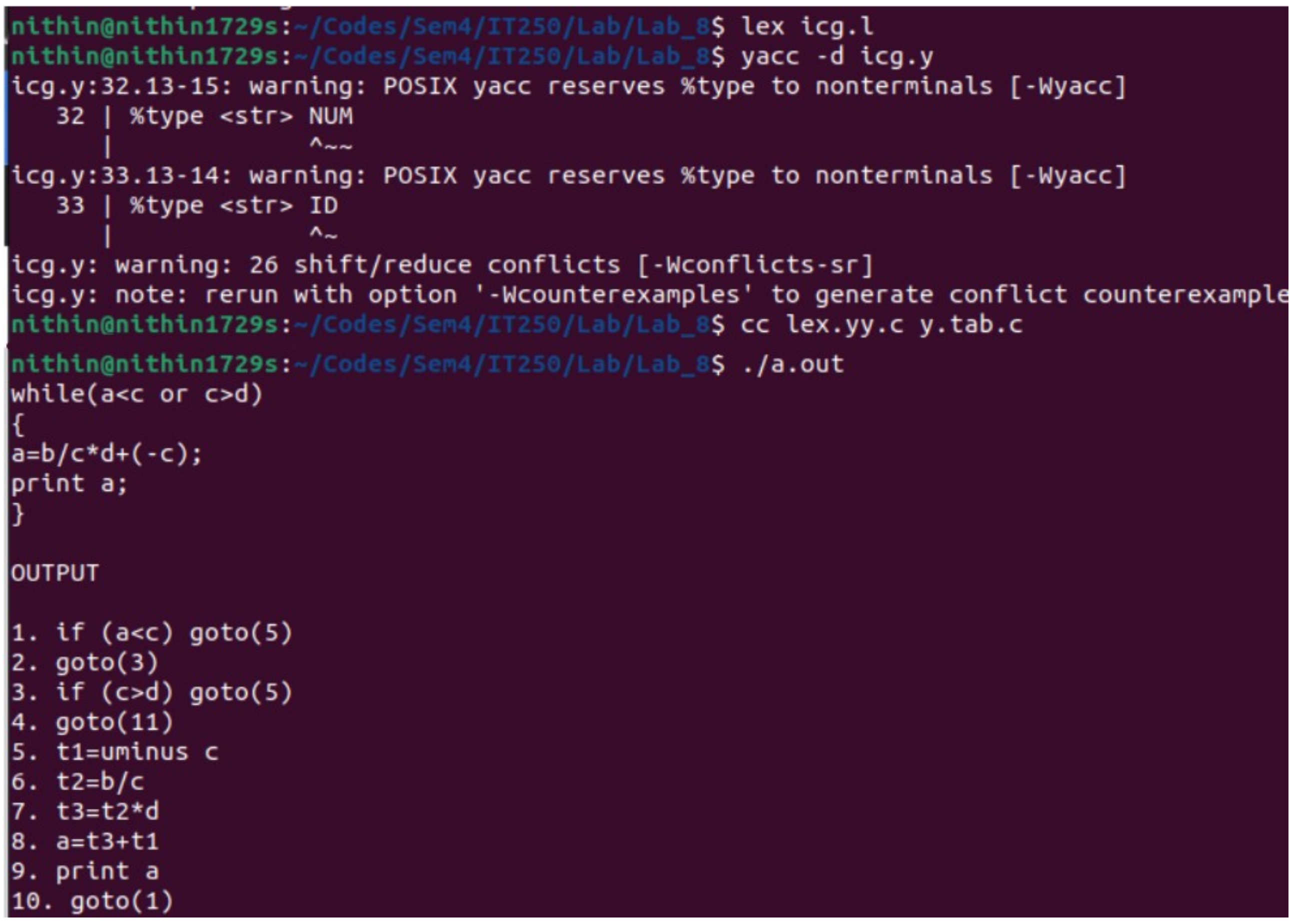


```
    countline++;
}
| EXPRN LE EXPRN {
    sprintf(ir, "%s\nt%d = %s <= %s", ir, countvar, $1, $3);
    $$[0] = '\0';
    sprintf($$, "t%d", countvar);
    if (ir[0] == '\n') {
        for (int i = 0; i < strlen(ir); i++) {
            ir[i] = ir[i + 1];
        }
    }
    countvar++;
    countline++;
}
| OP EXPRN CP {
    $$[0] = '\0';
    sprintf($$, "%s", $2);
}
| NUM {
    sprintf($$, "%s", $1);
}
| ID {
    sprintf($$, "%s", $1);
}
;

%%

int yyerror() {
    printf("\nParsing is failed.\n");
    return 0;
}

int main() {
    ir[0] = '\0';
    stack[0] = 1;
    yyparse();
    countline = 2;
    printf("1. ");
    for (int i = 0; i < strlen(ir); i++) {
        if (ir[i] == '\n') {
            printf("\n%d. ", countline);
            countline++;
        } else
            printf("%c", ir[i]);
    }
    printf("\n");
    return 0;
}
```

OUTPUT



```
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_8$ lex icg.l
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_8$ yacc -d icg.y
icg.y:32.13-15: warning: POSIX yacc reserves %type to nonterminals [-Wyacc]
   32 | %type <str> NUM
      |             ^~~
icg.y:33.13-14: warning: POSIX yacc reserves %type to nonterminals [-Wyacc]
   33 | %type <str> ID
      |             ^~
icg.y: warning: 26 shift/reduce conflicts [-Wconflicts-sr]
icg.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_8$ cc lex.yy.c y.tab.c
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_8$ ./a.out
a = 1;
b = 1;
while( a <= 5 )
{
b = 1;
while( b <= 5 )
{
b = b + 1;
print b;
}
a = a + 1;
print a;
}

OUTPUT

1. a = 1
2. b = 1
3. if (a<=5) goto(5)
4. goto(15)
5. b=1
6. if (b<=5) goto(8)
7. goto(11)
8. t1 = b + 1
9. b = t1
10. print b
11. t2 = a + 1
12. a = t2
13. print a
14. goto(3)
```



```
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_8$ lex icg.l
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_8$ yacc -d icg.y
icg.y:32.13-15: warning: POSIX yacc reserves %type to nonterminals [-Wyacc]
   32 | %type <str> NUM
      |             ^~~
icg.y:33.13-14: warning: POSIX yacc reserves %type to nonterminals [-Wyacc]
   33 | %type <str> ID
      |             ^~
icg.y: warning: 26 shift/reduce conflicts [-Wconflicts-sr]
icg.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexample
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_8$ cc lex.yy.c y.tab.c
nithin@nithin1729s:~/Codes/Sem4/IT250/Lab/Lab_8$ ./a.out
while(a<c or c>d)
{
a=b/c*d+(-c);
print a;
}

OUTPUT

1. if (a<c) goto(5)
2. goto(3)
3. if (c>d) goto(5)
4. goto(11)
5. t1=uminus c
6. t2=b/c
7. t3=t2*d
8. a=t3+t1
9. print a
10. goto(1)
```