Project Αρχές Γλωσσών Προγραμματισμού και Μεταφραστών

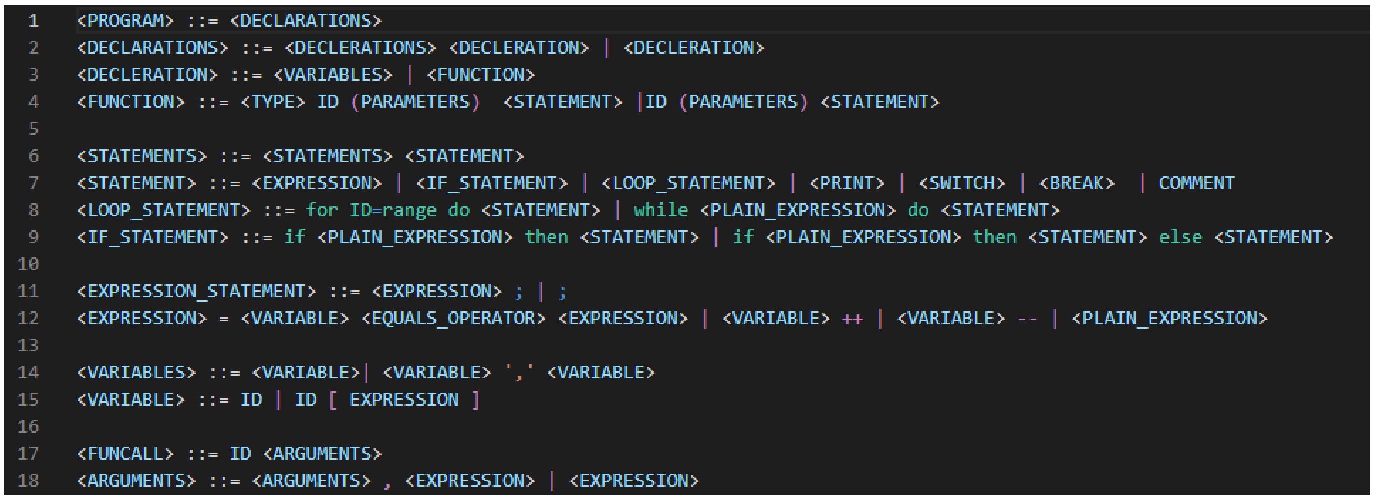
Στοιχεία Μελών

Ασημομύτης Δαμιανός, 1063427, 4ο, up1063427@upnet.gr

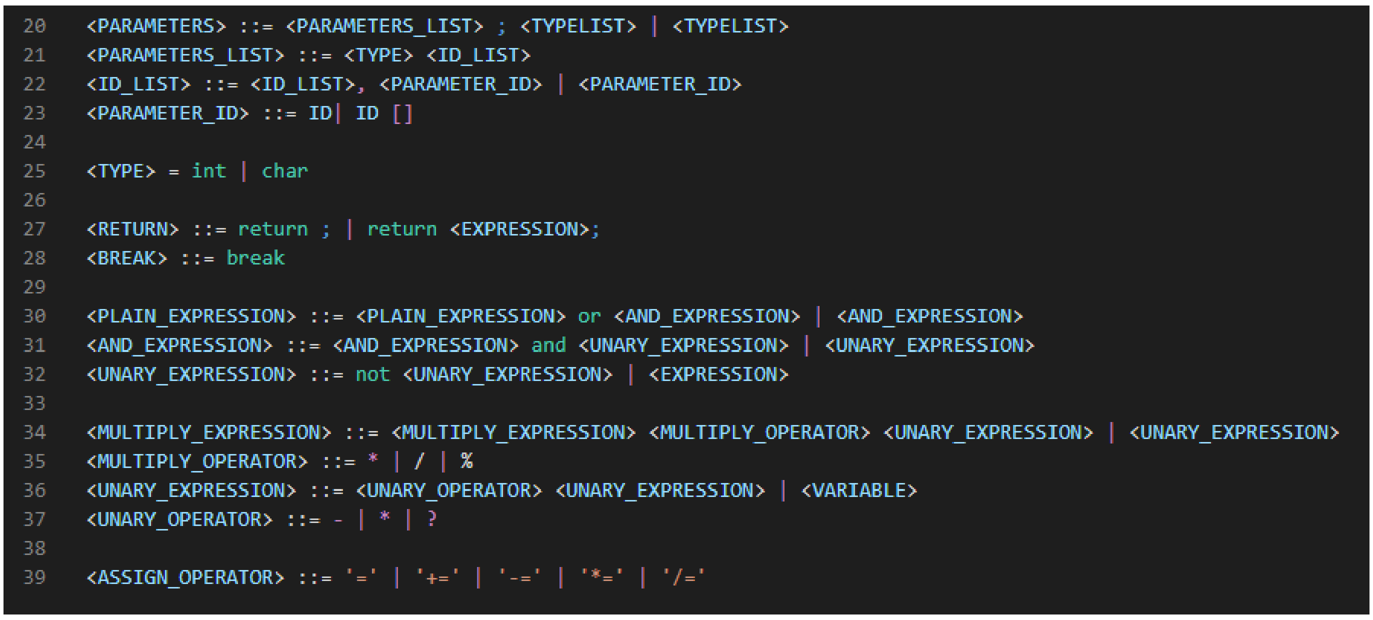
Βοντζαλίδης Γιώργος, 1047095, 6ο, up1047095@upnet.gr

Σφενδουράκης Παναγιώτης, 1054282, 5ο, sfendourakis@ceid.upatras.gr

BNF Γραμματική

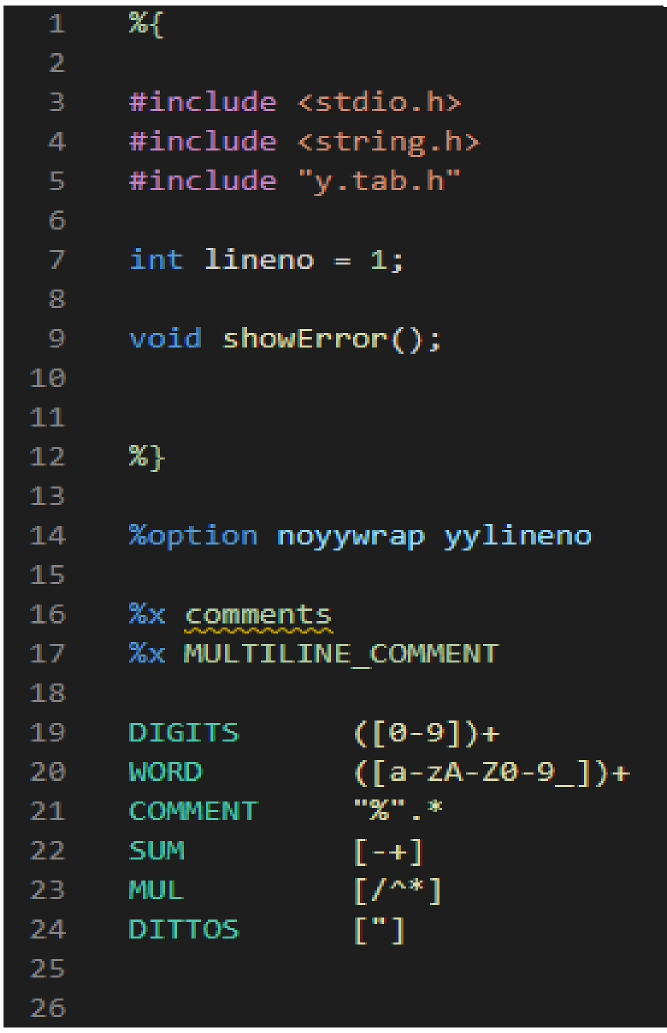


```
<PROGRAM> ::= <DECLARATIONS>
<DECLARATIONS> ::= <DECLERATIONS> <DECLERATION> | <DECLERATION>
<DECLERATION> ::= <VARIABLES> | <FUNCTION>
<FUNCTION> ::= <TYPE> ID (PARAMETERS)  <STATEMENT> |ID (PARAMETERS) <STATEMENT>

<STATEMENTS> ::= <STATEMENTS> <STATEMENT>
<STATEMENT> ::= <EXPRESSION> | <IF_STATEMENT> | <LOOP_STATEMENT> | <PRINT> | <SWITCH> | <BREAK>  | COMMENT
<LOOP_STATEMENT> ::= for ID=range do <STATEMENT> | while <PLAIN_EXPRESSION> do <STATEMENT>
<IF_STATEMENT> ::= if <PLAIN_EXPRESSION> then <STATEMENT> | if <PLAIN_EXPRESSION> then <STATEMENT> else <STATEMENT>

<EXPRESSION_STATEMENT> ::= <EXPRESSION> ; | ;
<EXPRESSION> = <VARIABLE> <EQUALS_OPERATOR> <EXPRESSION> | <VARIABLE> ++ | <VARIABLE> -- | <PLAIN_EXPRESSION>

<VARIABLES> ::= <VARIABLE>| <VARIABLE> ',' <VARIABLE>
<VARIABLE> ::= ID | ID [ EXPRESSION ]

<FUNCALL> ::= ID <ARGUMENTS>
<ARGUMENTS> ::= <ARGUMENTS> , <EXPRESSION> | <EXPRESSION>
```

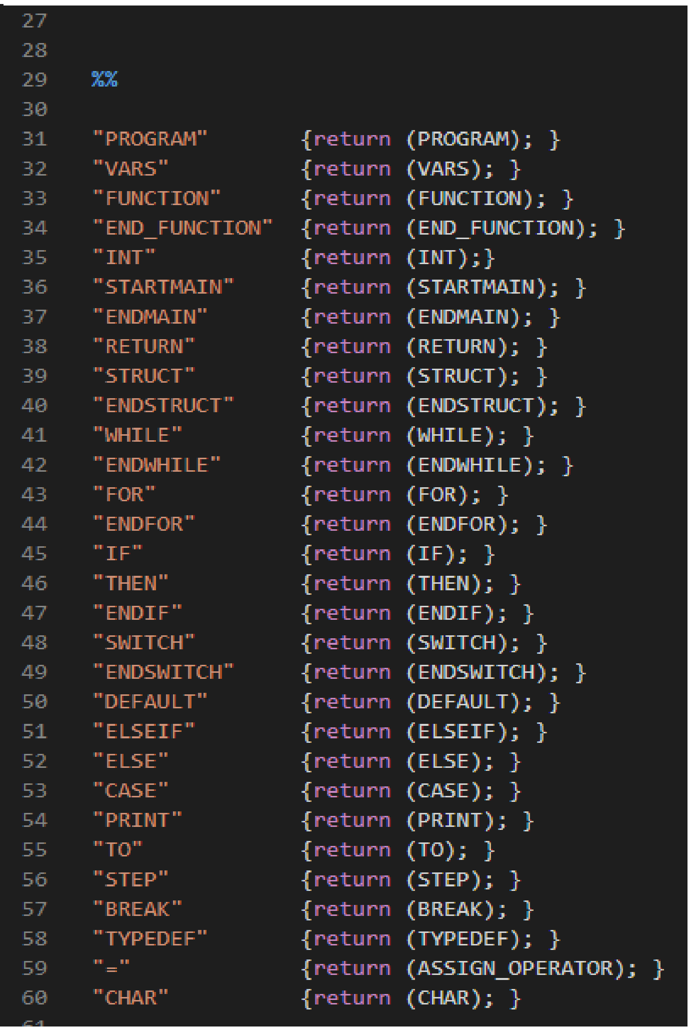


```
<PARAMETERS> ::= <PARAMETERS_LIST> ; <TYPELIST> | <TYPELIST>
<PARAMETERS_LIST> ::= <TYPE> <ID_LIST>
<ID_LIST> ::= <ID_LIST>, <PARAMETER_ID> | <PARAMETER_ID>
<PARAMETER_ID> ::= ID| ID []

<TYPE> = int | char

<RETURN> ::= return ; | return <EXPRESSION>;
<BREAK> ::= break

<PLAIN_EXPRESSION> ::= <PLAIN_EXPRESSION> or <AND_EXPRESSION> | <AND_EXPRESSION>
<AND_EXPRESSION> ::= <AND_EXPRESSION> and <UNARY_EXPRESSION> | <UNARY_EXPRESSION>
<UNARY_EXPRESSION> ::= not <UNARY_EXPRESSION> | <EXPRESSION>

<MULTIPLY_EXPRESSION> ::= <MULTIPLY_EXPRESSION> <MULTIPLY_OPERATOR> <UNARY_EXPRESSION> | <UNARY_EXPRESSION>
<MULTIPLY_OPERATOR> ::= * | / | %
<UNARY_EXPRESSION> ::= <UNARY_OPERATOR> <UNARY_EXPRESSION> | <VARIABLE>
<UNARY_OPERATOR> ::= - | * | ?

<ASSIGN_OPERATOR> ::= '=' | '+=' | '-=' | '*=' | '/='
```

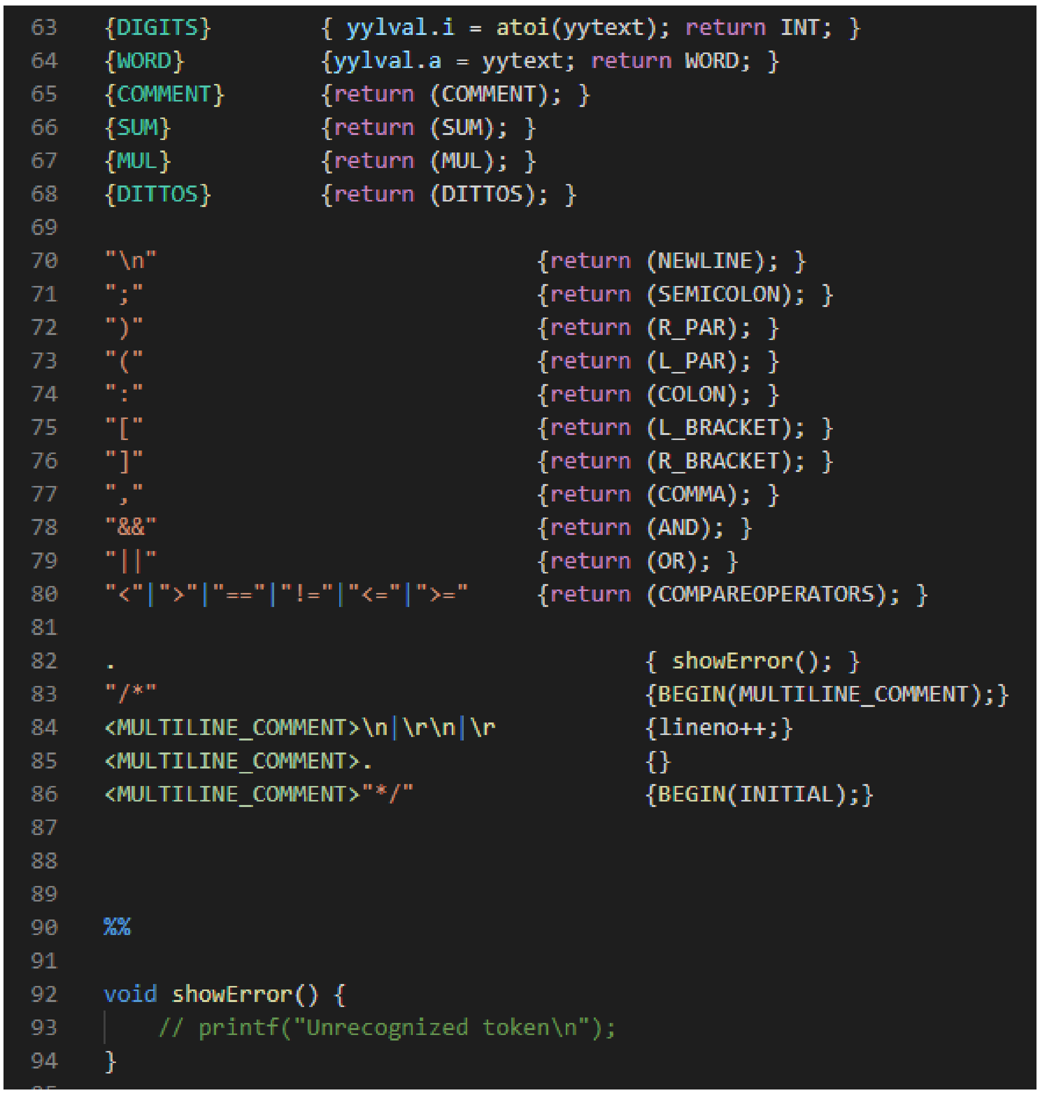
Αρχείο lexer.l



```
%{

#include <stdio.h>
#include <string.h>
#include "y.tab.h"

int lineno = 1;

void showError();


%}

%option noyywrap yylineno

%x comments
%x MULTILINE_COMMENT

DIGITS          ([0-9])+
WORD            ([a-zA-Z0-9_])+
COMMENT         "%".*
SUM             [-+]
MUL             [/^*]
DITTOS          ["]


```

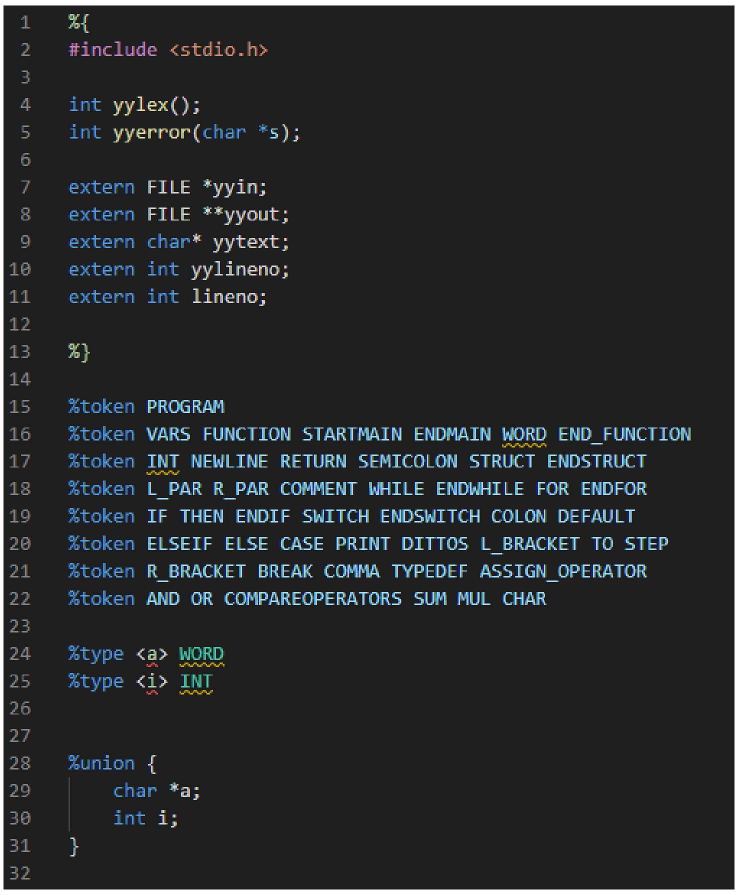


```


%%

"PROGRAM"       {return (PROGRAM); }
"VARS"          {return (VARS); }
"FUNCTION"      {return (FUNCTION); }
"END_FUNCTION"  {return (END_FUNCTION); }
"INT"           {return (INT);}
"STARTMAIN"     {return (STARTMAIN); }
"ENDMAIN"       {return (ENDMAIN); }
"RETURN"        {return (RETURN); }
"STRUCT"        {return (STRUCT); }
"ENDSTRUCT"     {return (ENDSTRUCT); }
"WHILE"         {return (WHILE); }
"ENDWHILE"      {return (ENDWHILE); }
"FOR"           {return (FOR); }
"ENDFOR"        {return (ENDFOR); }
"IF"            {return (IF); }
"THEN"          {return (THEN); }
"ENDIF"         {return (ENDIF); }
"SWITCH"        {return (SWITCH); }
"ENDSWITCH"     {return (ENDSWITCH); }
"DEFAULT"       {return (DEFAULT); }
"ELSEIF"        {return (ELSEIF); }
"ELSE"          {return (ELSE); }
"CASE"          {return (CASE); }
"PRINT"         {return (PRINT); }
"TO"            {return (TO); }
"STEP"          {return (STEP); }
"BREAK"         {return (BREAK); }
"TYPEDEF"       {return (TYPEDEF); }
"="             {return (ASSIGN_OPERATOR); }
"CHAR"          {return (CHAR); }
```

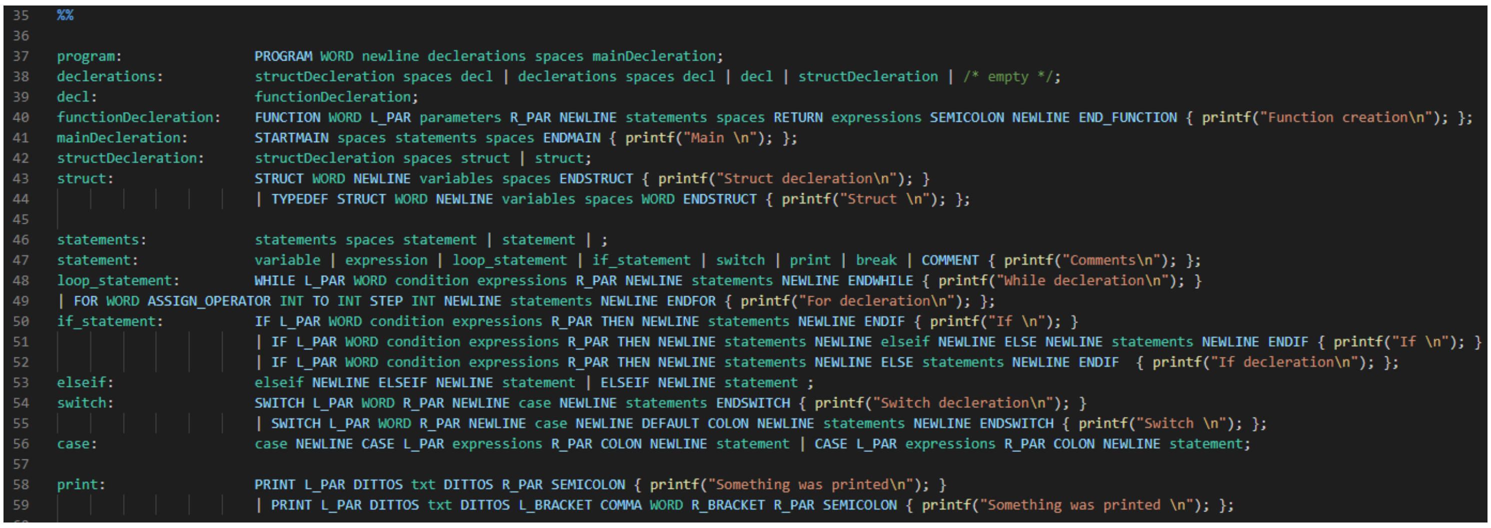


```
{DIGITS}        { yylval.i = atoi(yytext); return INT; }
{WORD}          {yylval.a = yytext; return WORD; }
{COMMENT}       {return (COMMENT); }
{SUM}           {return (SUM); }
{MUL}           {return (MUL); }
{DITTOS}        {return (DITTOS); }

"\n"                            {return (NEWLINE); }
";"                             {return (SEMICOLON); }
")"                             {return (R_PAR); }
"("                             {return (L_PAR); }
":"                             {return (COLON); }
"["                             {return (L_BRACKET); }
"]"                             {return (R_BRACKET); }
","                             {return (COMMA); }
"&&"                            {return (AND); }
"||"                            {return (OR); }
"<"|">"|"=="|"!="|"<="|">="     {return (COMPAREOPERATORS); }

.                                       { showError(); }
"/*"                                    {BEGIN(MULTILINE_COMMENT);}
<MULTILINE_COMMENT>\n|\r\n|\r           {lineno++;}
<MULTILINE_COMMENT>.                    {}
<MULTILINE_COMMENT>"*/"                 {BEGIN(INITIAL);}




%%

void showError() {
    // printf("Unrecognized token\n");
}
```

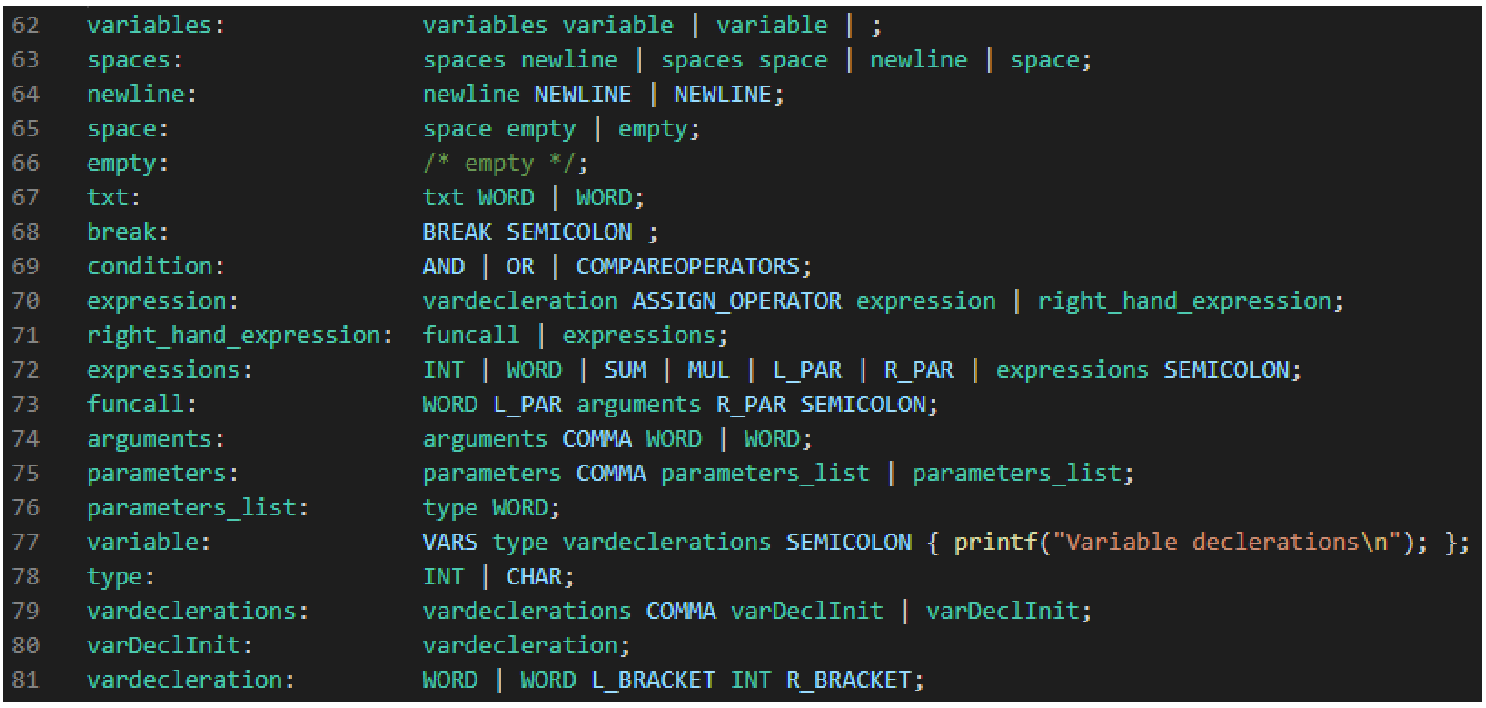
Αρχείο mybison.y



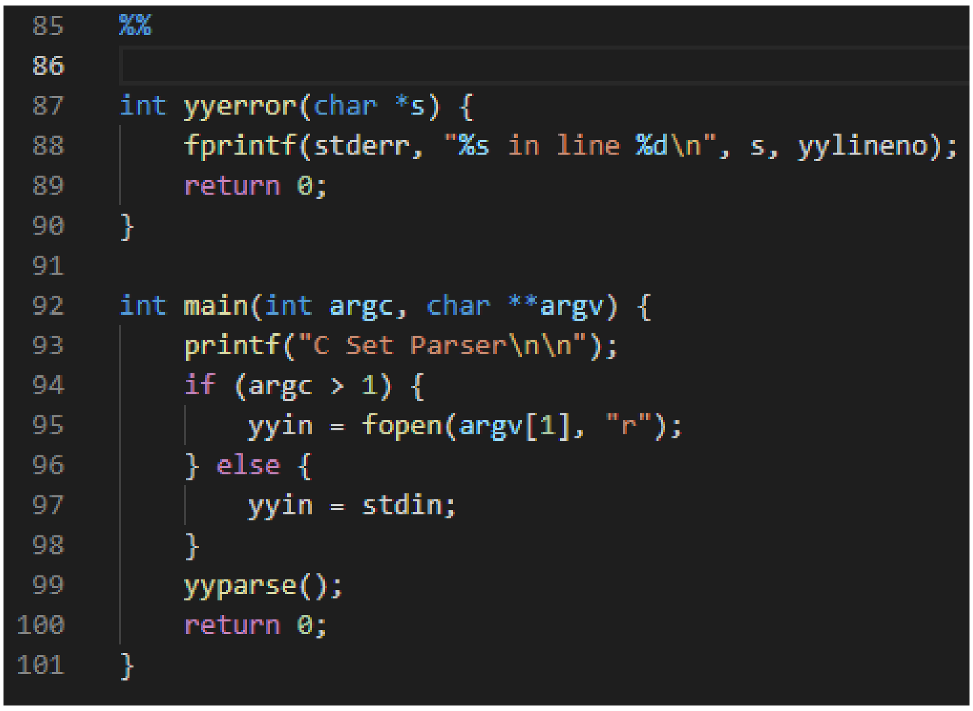
```
%{
#include <stdio.h>

int yylex();
int yyerror(char *s);

extern FILE *yyin;
extern FILE **yyout;
extern char* yytext;
extern int yylineno;
extern int lineno;

%}

%token PROGRAM
%token VARS FUNCTION STARTMAIN ENDMAIN WORD END_FUNCTION
%token INT NEWLINE RETURN SEMICOLON STRUCT ENDSTRUCT
%token L_PAR R_PAR COMMENT WHILE ENDWHILE FOR ENDFOR
%token IF THEN ENDIF SWITCH ENDSWITCH COLON DEFAULT
%token ELSEIF ELSE CASE PRINT DITTOS L_BRACKET TO STEP
%token R_BRACKET BREAK COMMA TYPEDEF ASSIGN_OPERATOR
%token AND OR COMPAREOPERATORS SUM MUL CHAR

%type <a> WORD
%type <i> INT


%union {
    char *a;
    int i;
}

```



```
%%

program:                PROGRAM WORD newline declerations spaces mainDecleration;
declerations:           structDecleration spaces decl | declerations spaces decl | decl | structDecleration | /* empty */;
decl:                   functionDecleration;
functionDecleration:    FUNCTION WORD L_PAR parameters R_PAR NEWLINE statements spaces RETURN expressions SEMICOLON NEWLINE END_FUNCTION { printf("Function creation\n"); };
mainDecleration:        STARTMAIN spaces statements spaces ENDMAIN { printf("Main \n"); };
structDecleration:      structDecleration spaces struct | struct;
struct:                 STRUCT WORD NEWLINE variables spaces ENDSTRUCT { printf("Struct decleration\n"); }
                        | TYPEDEF STRUCT WORD NEWLINE variables spaces WORD ENDSTRUCT { printf("Struct \n"); };

statements:             statements spaces statement | statement | ;
statement:              variable | expression | loop_statement | if_statement | switch | print | break | COMMENT { printf("Comments\n"); };
loop_statement:         WHILE L_PAR WORD condition expressions R_PAR NEWLINE statements NEWLINE ENDWHILE { printf("While decleration\n"); }
| FOR WORD ASSIGN_OPERATOR INT TO INT STEP INT NEWLINE statements NEWLINE ENDFOR { printf("For decleration\n"); };
if_statement:           IF L_PAR WORD condition expressions R_PAR THEN NEWLINE statements NEWLINE ENDIF { printf("If \n"); }
                        | IF L_PAR WORD condition expressions R_PAR THEN NEWLINE statements NEWLINE elseif NEWLINE ELSE NEWLINE statements NEWLINE ENDIF { printf("If \n"); }
                        | IF L_PAR WORD condition expressions R_PAR THEN NEWLINE statements NEWLINE ELSE statements NEWLINE ENDIF  { printf("If decleration\n"); };
elseif:                 elseif NEWLINE ELSEIF NEWLINE statement | ELSEIF NEWLINE statement ;
switch:                 SWITCH L_PAR WORD R_PAR NEWLINE case NEWLINE statements ENDSWITCH { printf("Switch decleration\n"); }
                        | SWITCH L_PAR WORD R_PAR NEWLINE case NEWLINE DEFAULT COLON NEWLINE statements NEWLINE ENDSWITCH { printf("Switch \n"); };
case:                   case NEWLINE CASE L_PAR expressions R_PAR COLON NEWLINE statement | CASE L_PAR expressions R_PAR COLON NEWLINE statement;

print:                  PRINT L_PAR DITTOS txt DITTOS R_PAR SEMICOLON { printf("Something was printed\n"); }
                        | PRINT L_PAR DITTOS txt DITTOS L_BRACKET COMMA WORD R_BRACKET R_PAR SEMICOLON { printf("Something was printed \n"); };
```

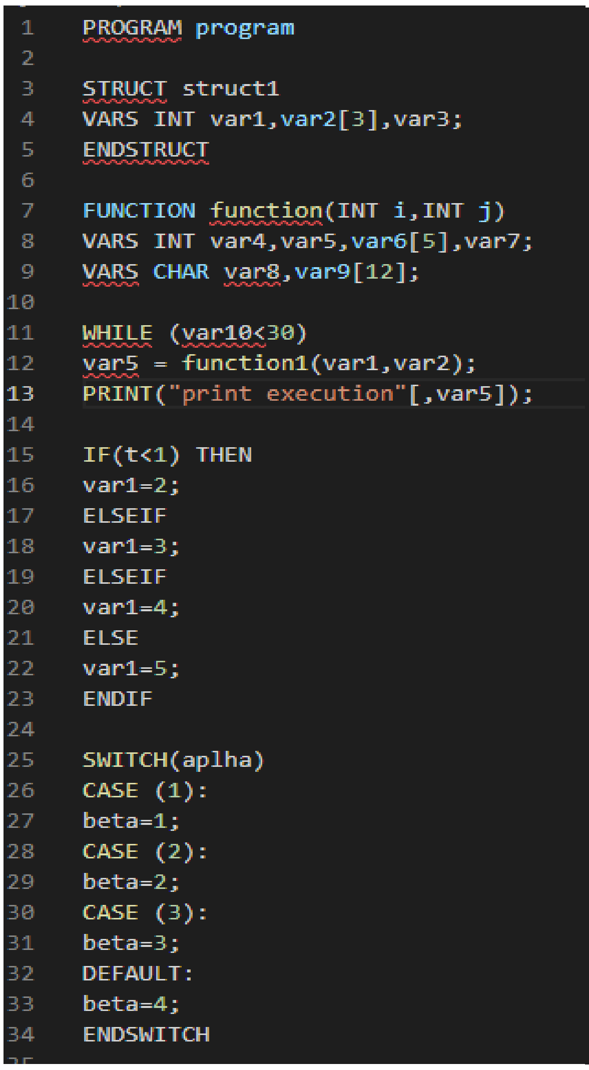


```
variables:              variables variable | variable | ;
spaces:                 spaces newline | spaces space | newline | space;
newline:                newline NEWLINE | NEWLINE;
space:                  space empty | empty;
empty:                  /* empty */;
txt:                    txt WORD | WORD;
break:                  BREAK SEMICOLON ;
condition:              AND | OR | COMPAREOPERATORS;
expression:             vardecleration ASSIGN_OPERATOR expression | right_hand_expression;
right_hand_expression:  funcall | expressions;
expressions:            INT | WORD | SUM | MUL | L_PAR | R_PAR | expressions SEMICOLON;
funcall:                WORD L_PAR arguments R_PAR SEMICOLON;
arguments:              arguments COMMA WORD | WORD;
parameters:             parameters COMMA parameters_list | parameters_list;
parameters_list:        type WORD;
variable:               VARS type vardeclerations SEMICOLON { printf("Variable declerations\n"); };
type:                   INT | CHAR;
vardeclerations:        vardeclerations COMMA varDeclInit | varDeclInit;
varDeclInit:            vardecleration;
vardecleration:         WORD | WORD L_BRACKET INT R_BRACKET;
```

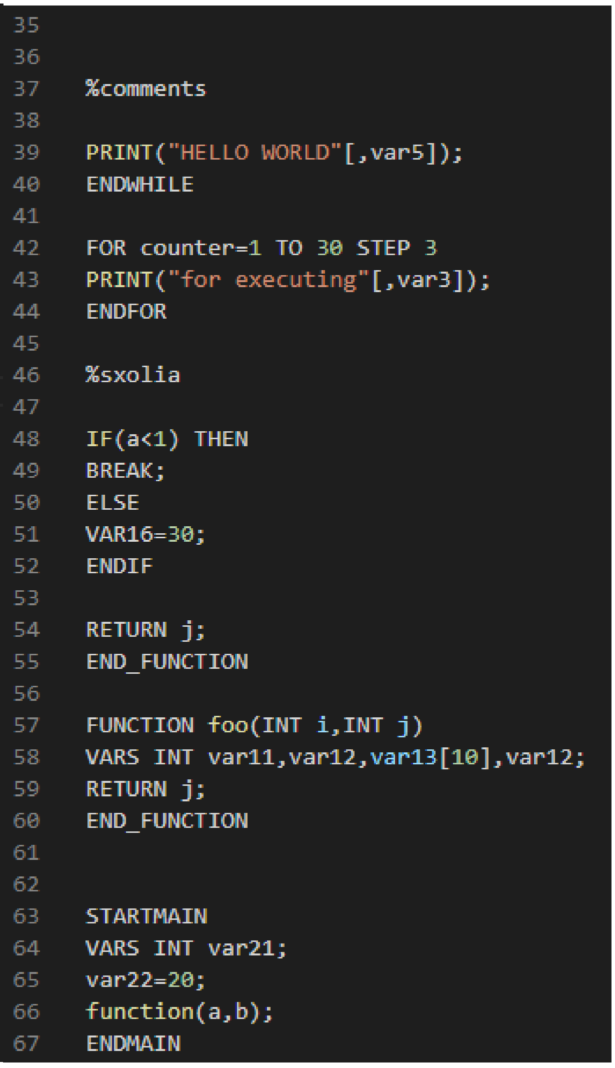


```
%%

int yyerror(char *s) {
    fprintf(stderr, "%s in line %d\n", s, yylineno);
    return 0;
}

int main(int argc, char **argv) {
    printf("C Set Parser\n\n");
    if (argc > 1) {
        yyin = fopen(argv[1], "r");
    } else {
        yyin = stdin;
    }
    yyparse();
    return 0;
}
```

Αρχείο C

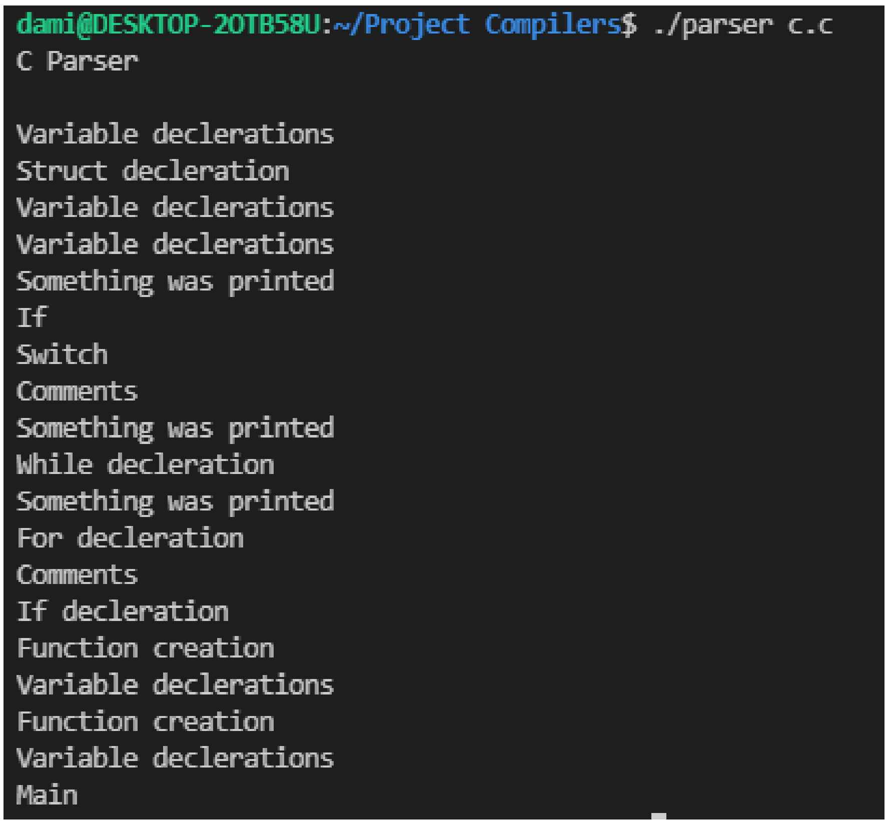


```
PROGRAM program

STRUCT struct1
VARS INT var1,var2[3],var3;
ENDSTRUCT

FUNCTION function(INT i,INT j)
VARS INT var4,var5,var6[5],var7;
VARS CHAR var8,var9[12];

WHILE (var10<30)
var5 = function1(var1,var2);
PRINT("print execution"[,var5]);

IF(t<1) THEN
var1=2;
ELSEIF
var1=3;
ELSEIF
var1=4;
ELSE
var1=5;
ENDIF

SWITCH(aplha)
CASE (1):
beta=1;
CASE (2):
beta=2;
CASE (3):
beta=3;
DEFAULT:
beta=4;
ENDSWITCH
```



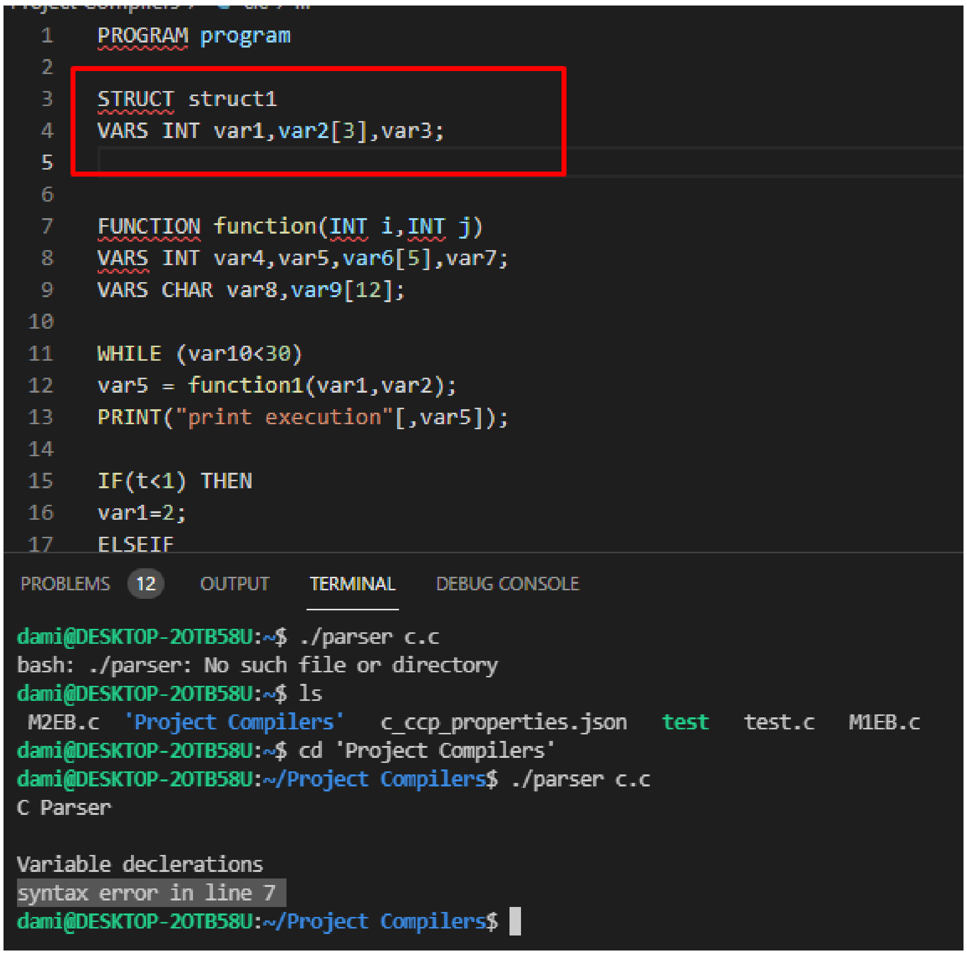
```


%comments

PRINT("HELLO WORLD"[,var5]);
ENDWHILE

FOR counter=1 TO 30 STEP 3
PRINT("for executing"[,var3]);
ENDFOR

%sxolia

IF(a<1) THEN
BREAK;
ELSE
VAR16=30;
ENDIF

RETURN j;
END_FUNCTION

FUNCTION foo(INT i,INT j)
VARS INT var11,var12,var13[10],var12;
RETURN j;
END_FUNCTION


STARTMAIN
VARS INT var21;
var22=20;
function(a,b);
ENDMAIN
```

Επιτυχής εκτέλεση του Parser



```
dami@DESKTOP-2OTB58U:~/Project Compilers$ ./parser c.c
C Parser

Variable declerations
Struct decleration
Variable declerations
Variable declerations
Something was printed
If
Switch
Comments
Something was printed
While decleration
Something was printed
For decleration
Comments
If decleration
Function creation
Variable declerations
Function creation
Variable declerations
Main
```

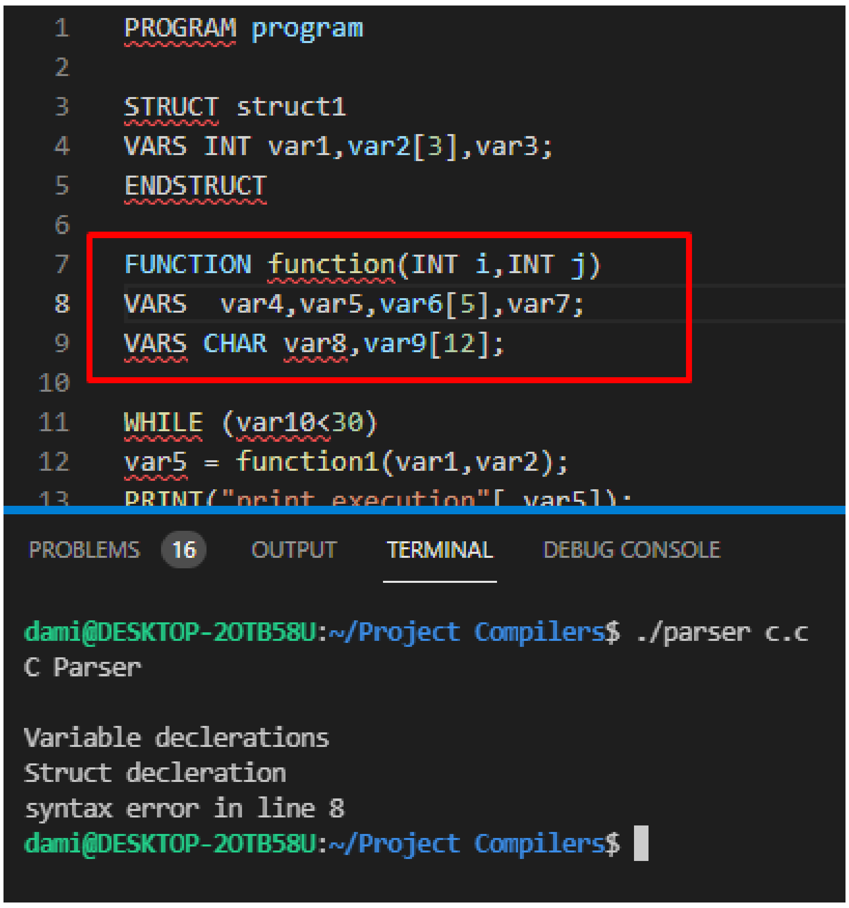
Μία λάθος δομή του STRUCT και το error κατά την εκτέλεση του parser:



```
 1  PROGRAM program
 2
 3  STRUCT struct1
 4  VARS INT var1,var2[3],var3;
 5
 6
 7  FUNCTION function(INT i,INT j)
 8  VARS INT var4,var5,var6[5],var7;
 9  VARS CHAR var8,var9[12];
10
11  WHILE (var10<30)
12  var5 = function1(var1,var2);
13  PRINT("print execution"[,var5]);
14
15  IF(t<1) THEN
16  var1=2;
17  ELSEIF
```

```
dami@DESKTOP-20TB58U:~$ ./parser c.c
bash: ./parser: No such file or directory
dami@DESKTOP-20TB58U:~$ ls
 M2EB.c  'Project Compilers'   c_ccp_properties.json   test   test.c   M1EB.c
dami@DESKTOP-20TB58U:~$ cd 'Project Compilers'
dami@DESKTOP-20TB58U:~/Project Compilers$ ./parser c.c
C Parser

Variable declerations
syntax error in line 7
dami@DESKTOP-20TB58U:~/Project Compilers$
```

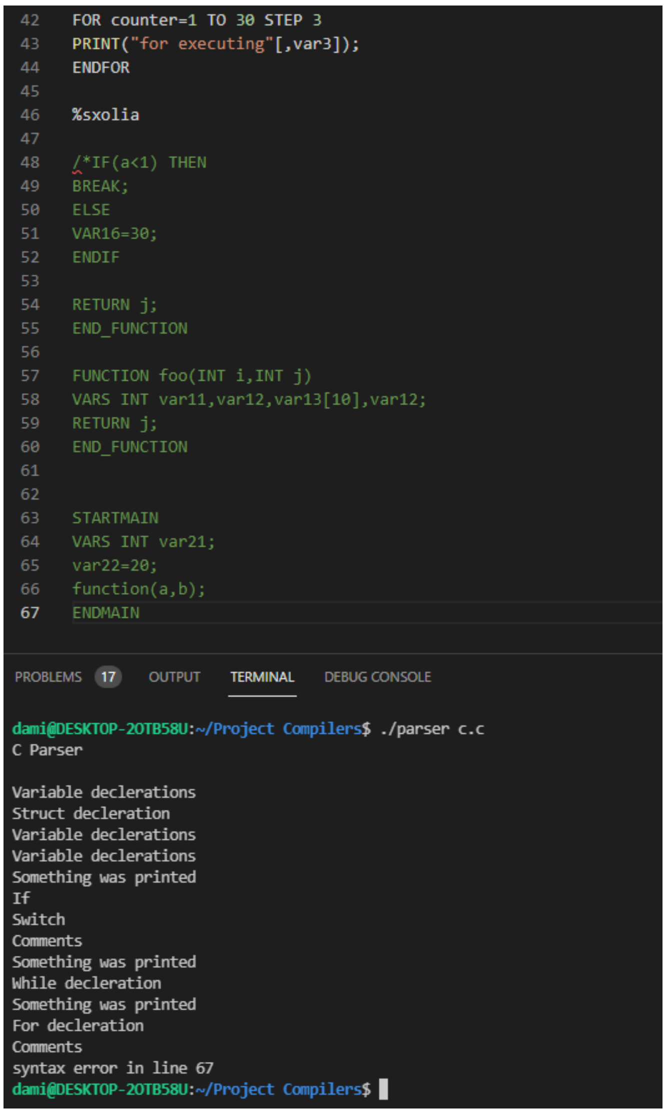
Λάθος δήλωση μεταβλητών και το αντίστοιχο error:



```
 1  PROGRAM program
 2
 3  STRUCT struct1
 4  VARS INT var1,var2[3],var3;
 5  ENDSTRUCT
 6
 7  FUNCTION function(INT i,INT j)
 8  VARS  var4,var5,var6[5],var7;
 9  VARS CHAR var8,var9[12];
10
11  WHILE (var10<30)
12  var5 = function1(var1,var2);
13  PRINT("print execution"[,var5]);
```

```
dami@DESKTOP-20TB58U:~/Project Compilers$ ./parser c.c
C Parser

Variable declerations
Struct decleration
syntax error in line 8
dami@DESKTOP-20TB58U:~/Project Compilers$
```

Λάθος χρήση multiline comment και το αντίστοιχο error:



```
42   FOR counter=1 TO 30 STEP 3
43   PRINT("for executing"[,var3]);
44   ENDFOR
45
46   %sxolia
47
48   /*IF(a<1) THEN
49   BREAK;
50   ELSE
51   VAR16=30;
52   ENDIF
53
54   RETURN j;
55   END_FUNCTION
56
57   FUNCTION foo(INT i,INT j)
58   VARS INT var11,var12,var13[10],var12;
59   RETURN j;
60   END_FUNCTION
61
62
63   STARTMAIN
64   VARS INT var21;
65   var22=20;
66   function(a,b);
67   ENDMAIN
```

```
dami@DESKTOP-2OTB58U:~/Project Compilers$ ./parser c.c
C Parser

Variable declerations
Struct decleration
Variable declerations
Variable declerations
Something was printed
If
Switch
Comments
Something was printed
While decleration
Something was printed
For decleration
Comments
syntax error in line 67
dami@DESKTOP-2OTB58U:~/Project Compilers$
```