# ΑΡΧΕΣ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

ΕΞΑΜΗΝΙΑΙΑ ΕΡΓΑΣΙΑ ΣΤΟ ΜΑΘΗΜΑ

“ΑΡΧΕΣ ΓΛΩΣΣΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ”

ΙΩΑΝΝΗΣ ΚΟΛΛΙΑΣ
ΑΜ: 1064886
Έτος: 4
email: **up1064886@upnet.gr**

ΚΩΝΣΤΑΝΤΙΝΟΣ ΞΗΡΟΓΙΑΝΝΗΣ
ΑΜ: 1047186
Έτος: 6
email: **xirogiannis@ceid.upatras.gr**

ΑΝΔΡΕΑΣ ΤΣΙΡΩΝΗΣ
ΑΜ: 1063428
Έτος: 4
email: **andretsironis55@gmail.com**

# ΕΙΣΑΓΩΓΗ

Στην εργασία μας, δημιουργήσαμε μέσω των flex και bison, ένα scanner και parser αντίστοιχα, όπου αναγνωρίζουν μια ψευδογλώσσα, της οποίας οι κανόνες της συντακτικής γραμματικής έχουν οριστεί στην εκφώνηση της εργασίας. Έχει πραγματοποιηθεί το βασικό ερώτημα της άσκησης, όπως και τα ερωτήματα που σχετίζονταν με την δημιουργία δήλωσης μεταβλητής Struct πριν το πρόγραμμα και σχόλια πολλαπλών μεταβλητών χωρίς χρήση κανονικής έκφρασης.

Επίσης, ο parser δεν τρέχει όντως το πρόγραμμα, απλώς εξετάζει αν το πρόγραμμα έχει γραφτεί με σωστό τρόπο. Από την flex γυρνάει τα περισσότερα token χωρίς τα semantic value, καθώς δεν χρειαζόταν να υπολογίσει ή να θυμάται μεταβλητές. Όλες οι δεσμευμένες λέξεις είναι υποχρεωτικά κεφαλαία. Όλα τα Terminal token έχουν T_ μπροστά, για να ξεχωρίζουν σαν Token. Στους κανόνες, χρησιμοποιούνται τα ονόματα των token όπως γυρίστηκαν από το flex, αλλά έχουμε στα %token της bison σε ποιο σύμβολο αντιστοιχούν.

# BNF GRAMMAR

Η BNF γραμματική έχει ως εξής:

```
start::= T_PROGRAM  T_ID T_PROGRAM_LINE program

program::= Structs functions main


Structs::=struct Structs

struct::= T_TYPEDEF T_STRUCT T_ID T_STRUCT_LINE T_VARS structdeclare T_ID T_ENDSTRUCT
    |  T_STRUCT T_ID T_STRUCT_LINE T_VARS structdeclare  T_ENDSTRUCT

structdeclare::=T_INT T_ID T_SEMI structdeclare| T_CHAR T_ID T_SEMI structdeclare | T_INT T_ID
T_LBRACK T_ICONST T_RBRACK T_SEMI  structdeclare
    | T_CHAR T_ID T_LBRACK T_ICONST T_RBRACK T_SEMI structdeclare

functions::= function functions

function::= T_FUNCTION T_ID T_LPAREN parameters T_RPAREN body return T_ENDFUNCTION

        |T_FUNCTION T_ID T_LPAREN T_RPAREN body return T_ENDFUNCTION

parameters::= parameter T_COMA parameters
| parameter

parameter::= T_INT Variable
|
T_CHAR Variable

Variable::= T_ID
|
T_ID T_LBRACK T_ICONST T_RBRACK


body::= T_VARS Declarations  commands

commands::=  Assigns commands |If commands|Switch commands|Print commands|while commands|For co
mmands

calc::= T_ADDOP | T_MULOP

Assigns::= parameter T_ASSIGN expr T_SEMI | Variable T_ASSIGN expr  T_SEMI

expr::=T_ICONST | T_ID | expr calc expr | T_ADDOP expr NEG |  expr T_POWER expr | T_LPAREN expr
 T_RPAREN  | functionassign

functionassign::=T_ID T_LPAREN parametersassign T_RPAREN

parametersassign::=Variable T_COMA Variable


while::= T_WHILE T_LPAREN condition T_RPAREN  bodywhile break T_ENDWHILE

bodywhile::= commands | break commands
```

```
break::=T_BREAK T_SEMI


If::= T_IF T_LPAREN condition T_RPAREN T_THEN commands ELSEIF ELSE T_ENDIF

condition::= T_ID T_EQUOP T_ID | T_ID T_RELOP T_ID | T_ID T_ANDOP T_ID | T_ID T_OROP T_ID |T_IC
ONST

ELSEIF::=T_ELSEIF T_LPAREN condition T_RPAREN commands ELSEIF

ELSE::=T_ELSE commands

Switch::=T_SWITCH T_LPAREN T_ID T_RPAREN CASE DEFAULT T_ENDSWITCH

CASE::=T_CASE T_LPAREN T_ID T_RPAREN T_COLON commands CASE |T_CASE T_LPAREN T_ICONST T_RPAREN T
_COLON commands CASE | T_CASE T_LPAREN T_CCONST T_RPAREN T_COLON commands CASE

DEFAULT::=T_DEFAULT T_COLON commands

Print::=T_PRINT T_LPAREN  TEXT  VAR T_RPAREN T_SEMI

TEXT::=T_TEXT

For::=T_FOR T_COUNTER T_ICONST T_TO T_ICONST T_STEP T_ICONST bodywhile break T_ENDFOR

VAR::= T_LBRACK T_COMA T_ID T_RBRACK

Declarations::=  Declaration T_SEMI Declarations | Assigns
Declaration::=T_INT Variables |  T_CHAR Variables

Variables::= Variable T_COMA Variables | Variable

main::= T_STARTMAIN body return T_ENDMAIN

return::= T_RETURN T_ID T_SEMI |T_RETURN T_ICONST T_SEMI | T_RETURN T_CCONST T_SEMI
```

# FLEX

Ο λεκτικός μας αναλυτής φαίνεται παρακάτω:

```
%{
#include <stdio.h>
#include "parser.tab.h"
#define YY_USER_ACTION                                                \
  yylloc.first_line = yylloc.last_line;                               \
  yylloc.first_column = yylloc.last_column;                           \
  if (yylloc.last_line == yylineno)                                   \
    yylloc.last_column += yyleng;                                     \
  else {                                                              \
    yylloc.last_line = yylineno;                                      \
    yylloc.last_column = yytext + yyleng - strrchr(yytext, '\n');   \
  }


int lineno=1;
int error_count=0;

%}

/* Μεταβλητή τύπου int ενσωματωμένη στο Flex. Κάθε φορά που θα συναντά το Flex τον
χαρακτήρα νέας γραμμής ('\n'), η μεταβλητή θα αυξάνεται ΑΥΤΟΜΑΤΑ κατά 1 */
%option yylineno

/* Για την ανάγνωση ΜΟΝΟ ενός αρχείου κάθε φορά */
%option noyywrap


%x PROGRAM
%x STRUCT

LETTER      [a-zA-Z]
DIGIT       [0-9]
CHARACTER   {LETTER}|{DIGIT}
NZNUMBER    [1-9]{DIGIT}*|0
ID          _?{LETTER}({LETTER}|{DIGIT}|_)*
ICONST      {NZNUMBER}
FCONST      {NZNUMBER}\.{DIGIT}*
CCONST      \'.\'
WHITESPACE  [ \t\t\r]+
TEXT        \"(\\.|[^"\\])*\"
COMMENT     %(.)*(\r\n|\r|\n)
```

```
%%

<PROGRAM>\r\n|\r|\n {printf("Found KEYWORD ->  PROGRAM_CHANGE_LINE\n"); BEGIN (INITIAL);
return T_PROGRAM_LINE; }
<STRUCT>\r\n|\r|\n  {printf("Found KEYWORD ->  CHANGE_LINE\n"); BEGIN (INITIAL); return
T_STRUCT_LINE; }
"STRUCT"            { printf("Found KEYWORD -> STRUCT\n"); BEGIN (STRUCT); return
T_STRUCT; }
"ENDSTRUCT"         { printf("Found KEYWORD -> ENDSTRUCT\n"); return T_ENDSTRUCT; }
"VARS"              { printf("Found KEYWORD -> VARS\n"); return T_VARS; }
"TYPEDEF"           { printf("Found KEYWORD -> TYPEDEF\n"); return T_TYPEDEF; }
"/*"                { comment(); }
"PROGRAM"           { printf("Found KEYWORD -> program\n");BEGIN (PROGRAM); return
T_PROGRAM; }
"CHAR"              { printf("Found KEYWORD -> char\n"); return T_CHAR; }
"INT"               { printf("Found KEYWORD -> int\n"); return T_INT; }
"FUNCTION"          { printf("Found KEYWORD -> function\n"); return T_FUNCTION; }
"END_FUNCTION"      { printf("Found KEYWORD -> end_function\n"); return T_ENDFUNCTION ; }
"IF"                { printf("Found KEYWORD -> if\n"); return T_IF; }
"THEN"              { printf("Found KEYWORD -> then\n"); return T_THEN; }
"ELSEIF"            { printf("Found KEYWORD -> elseif\n"); return T_ELSEIF; }
"ELSE"              { printf("Found KEYWORD -> else\n"); return T_ELSE; }
"ENDIF"             { printf("Found KEYWORD -> endif\n"); return T_ENDIF; }
"SWITCH"              { printf("Found KEYWORD -> switch\n"); return T_SWITCH; }
"CASE"               { printf("Found KEYWORD -> case\n"); return T_CASE; }
"DEFAULT"            { printf("Found KEYWORD -> default\n"); return T_DEFAULT; }
"ENDSWITCH"          { printf("Found KEYWORD -> endswitch\n"); return T_ENDSWITCH; }
"BREAK"              { printf("Found KEYWORD -> break\n"); return T_BREAK; }
"WHILE"             { printf("Found KEYWORD -> while\n"); return T_WHILE; }
"ENDWHILE"           { printf("Found KEYWORD -> endwhile\n"); return T_ENDWHILE; }
"FOR"               { printf("Found KEYWORD -> case\n"); return T_FOR; }
"TO"               { printf("Found KEYWORD -> case\n"); return T_TO; }
"STEP"               { printf("Found KEYWORD -> case\n"); return T_STEP; }
"ENDFOR"               { printf("Found KEYWORD -> case\n"); return T_ENDFOR; }
"PRINT"              { printf("Found KEYWORD -> PRINT\n"); return T_PRINT; }
"RETURN"            { printf("Found KEYWORD -> return\n"); return T_RETURN; }
"STARTMAIN"         { printf("Found KEYWORD-> startmain\n"); return T_STARTMAIN; }
"ENDMAIN"             { printf("Found KEYWORD -> endmain\n"); return T_ENDMAIN; }
"OR"                { printf("Found OROP\n"); return T_OROP; }
"AND"                { printf("Found ANDOP\n"); return T_ANDOP; }
"=="|"!="           { printf("Found EQUOP -> %s\n", yytext); return T_EQUOP; }
">"|"<"               { printf("Found RELOP -> %s\n", yytext); return T_RELOP; }
"+"|"-"             { printf("Found ADDOP -> %s\n", yytext); return T_ADDOP; }
"*"|"/"|"%"         { printf("Found MULOP -> %s\n", yytext); return T_MULOP; }
"!"                 { printf("Found NOTOP\n"); return T_NOTOP; }
"("                 { printf("Found LPAREN\n"); return T_LPAREN; }
")"                 { printf("Found RPAREN\n"); return T_RPAREN; }
";"                 { printf("Found SEMI\n"); return T_SEMI; }
","                 { printf("Found COMMA\n"); return T_COMA; }
"="                 { printf("Found ASSIGN\n"); return T_ASSIGN; }
```

```
"["                 { printf("Found LBRACK\n"); return T_LBRACK; }
"]"                 { printf("Found RBRACK\n"); return T_RBRACK; }
":"                  { printf("Found COLON\n"); return T_COLON; }
"COUNTER:="           { printf("Found KEYWORD -> COUNTER \n"); return T_COUNTER; }
"^"                   { printf("Found POWER\n"); return T_POWER; }
{COMMENT}           { printf("Found comment -> %s\n", yytext); /* ta comments ta opoia
ginontai ignore */ }
<STRUCT,INITIAL>{ID}   { yylval.charval= strdup(yytext);  printf("Found KEYWORD -> ID
\n"); return T_ID; }
<PROGRAM,INITIAL>{ID} { yylval.charval= strdup(yytext);  printf("Found KEYWORD -> ID
\n"); return T_ID; }

{TEXT}              { printf("Found TEXT -> %s\n", yytext); return T_TEXT; }
{ICONST}            { printf("Found ICONST -> %s\n", yytext); return T_ICONST; }
{CCONST}            { printf("Found CCONST -> %s\n", yytext); return T_CCONST; }
{WHITESPACE}        { /* Κενά (space) μέσα στο αρχείο - απλά τα αγνοούμε */ }
\n                  { lineno++; /* Εναλλακτικά βασιζόμαστε στο Flex να αυξήσει την μεταβλητή
'yylineno' κατά 1 AYTOMATA */ }
<<EOF>>             {printf("we reached the end of file (EOF)!\n"); return 0;}
.                   { error_count++; printf("\nUnrecognised character at line %d->%s!\n",
yylineno,yytext); }

%%

comment()
{
    char c, c1;

loop:
    while ((c = input()) != '*' && c != 0)
        putchar(c);

    if ((c1 = input()) != '/' && c != 0)
    {
        unput(c1);
        goto loop;
    }

    if (c != 0)
        putchar(c1);
}
```

Αρχικά ξεκινάμε τον λεκτικό μας αναλυτή κάνοντας include τις απαραίτητες βιβλιοθήκες αλλά και τη βιβλιοθήκη που δημιουργεί ο bison για τα tokens.

Έπειτα ακολουθεί define YY_USER_ACTION. Το define αυτό χρησιμοποιείται έτσι ώστε μετά το όνομά της FUNCTION να ακολουθεί αναγκαστικά αλλαγή γραμμής. Σημαντικό ρόλο στο define αυτό έχει strrchr μιας και γυρνάει έναν pointer για τον τελευταίο χαρακτήρα του string.

Ορίζουμε error_count ως μετρητής λαθών στο πρόγραμμα που ελέγχεται και μ' αυτό κλείνουν οι βιβλιοθήκες και οι δηλώσεις.

Συνεχίζοντας χρησιμοποιούμε διάφορες ρυθμίσεις της flex όπως το %option yylineno ( μεταβλητή int ενσωματωμένη στο flex, η οποία κάθε φορά που συναντά αλλαγή γραμμής αυξάνεται κατά 1) και το %option noyywrap ( για ανάγνωση μόνο ενός αρχείου τη φορά ). %x PROGRAM και %x STRUCT χρησιμοποιούνται για την αναγκαστική αλλαγή γραμμής μετά την εμφάνισή τους.

Ακολουθούν οι κανόνες του λεκτικού αναλυτή με τα διάφορα regex και τις επιστροφές τιμών.  Αξίζει να σταθούμε στα <PROGRAM>\r\n|\r|\n και <STRUCT>\r\n|\r|\n, όπου μετά από αυτά τα token, είναι αναγκαία η αλλαγή γραμμής. Αλλά επειδή δεν θέλουμε να μετράμε κάθε αλλαγή γραμμής token γιατί θα είχαμε θέματα με τον κώδικα, βάζουμε την flex να μπει στις ειδικές καταστάσεις που έχουμε ορίσει, ωστέ να γυρνάει token αλλαγή γραμμής μόνο σε αυτή την περίπτωση και μετά να συνεχίζει στην κανονική λειτουργία προγράμματος. Αυτό γίνεται στην PROGRAM και STRUCT, οπού το μόνο άλλο token που υπάρχει είναι το T_ID, όπου έχει συμπεριληφθεί να χρησιμοποιείται και σε αυτές τις ειδικές περιπτώσεις. Στην function οπού είχε περισσότερα token και μια τέτοια πρακτική θα ήταν μη πρακτική, χρησιμοποιήσαμε locations.

Κάθε token που βρίσκουμε,  τυπώνεται το όνομα του token και μετά κάνει return στην bison το token που αναγνωρίστηκε . Η T_ID  είναι η μόνη που γυρνάει και το semantic values, επειδή είναι αναγκαία λόγω του κανόνα struct. Κάνουμε printf τα token που αναγνωρίστηκαν για λόγω ευκολίας debugging.

Επεξηγηση κάποιων κανονικών εκφράσεων που χρησιμοποιήσουμε:

NZNUMBER : Ακέραιος αριθμός, οπού δεν έχει μηδέν μπροστά

ID  : Όνομα μεταβλητης ή όνομα functions ή όνομα προγράμματος, όπου σαν αρχικό σύμβολο πρέπει να είναι γράμμα και μετά μπορεί να είναι γράμμα, αριθμός ή κάτω παύλα (_)

ICONST  : Ακέραια σταθερή (ουσιαστικά int)

CCONST  : Ουσιαστικά ένα οποιοδήποτε σύμβολο.

COMMENT : Για σχόλιο μιας γραμμής

TEXT : Για την printf

Τέλος το multi-line comment πετυχαίνεται μέσω συνάρτησης που φαίνεται στο τρίτο και τελευταίο μέρος του λεκτικού αναλυτή.

# BISON

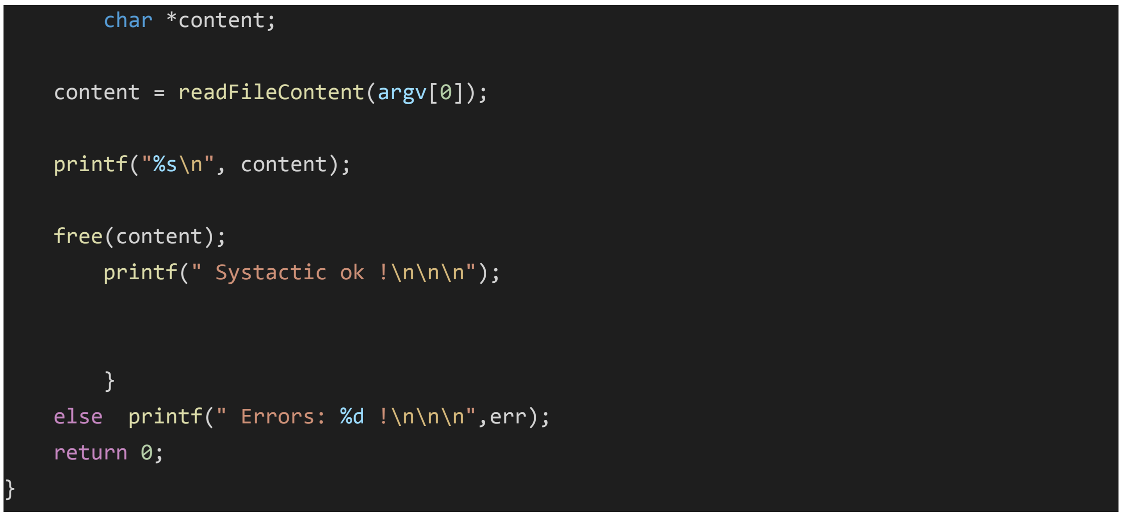
Ο συντακτικός μας αναλυτής έχει ως εξής:

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


extern FILE *yyin;
extern FILE *yyout;
void yyerror(const char *s);
int yylex();
extern int yylineno;
extern int lineno;

int err=0;


%}
%locations
%union{
       int intval;
           char* charval;

}
%token T_PROGRAM_LINE "to change line meta to onoma tou program."
%token  T_COLON  ":"
%token  T_LPAREN "("
%token  T_RPAREN ")"
%token  T_SEMI ";"
%token  T_COMA ","
%token  T_ASSIGN "="
%token  T_LBRACK "["
%token  T_RBRACK "]"
%token T_STRUCT "Struct"
%token T_VARS "VARS"
%token T_ENDSTRUCT "ENDSTRUCT"
%token T_TYPEDEF    "TYPEDEF"
%token T_STRUCT_LINE "to struct mazi me line change"
%token  T_PROGRAM T_FUNCTION T_ENDFUNCTION T_STARTMAIN T_ENDMAIN T_CHAR T_INT T_IF T_THEN
T_ELSEIF T_ELSE T_ENDIF T_ANDOP T_OROP T_RETURN T_WHILE T_ENDWHILE T_SWITCH T_CASE
T_DEFAULT T_ENDSWITCH T_BREAK T_FOR T_ENDFOR T_STEP T_COUNTER T_TO T_PRINT
%token <charval> T_ID T_CCONST
%token <intval> T_ICONST
%token T_POWER "^"
```

```
%token T_TEXT "otidhpote mesa se eisagwgika"
%token  T_RELOP "<, >, <=, >="
%token  T_EQUOP "=="
%token  T_NOTOP "!="
%token  T_ADDOP "+, -"
%token  T_MULOP "*, /, %"
%precedence NEG
%precedence T_THEN
%precedence T_ELSE
%left T_ANDOP T_OROP
%left T_RELOP T_EQUOP T_NOTOP
%left T_ADDOP
%left T_MULOP

%%

start: T_PROGRAM  T_ID T_PROGRAM_LINE program ;

program: Structs functions main ;

Structs:struct Structs | %empty;

    struct: T_TYPEDEF T_STRUCT T_ID T_STRUCT_LINE T_VARS structdeclare T_ID T_ENDSTRUCT
    {char* lathos ="lathos onoma" ; int value = strcmp ( $3,$7); if(value!=0)
{yyerror(lathos); printf("Λάθος όνομα"); YYABORT; } }
    |  T_STRUCT T_ID T_STRUCT_LINE T_VARS structdeclare    T_ENDSTRUCT   ;



    structdeclare:T_INT T_ID T_SEMI structdeclare| T_CHAR T_ID T_SEMI structdeclare |
T_INT T_ID "["T_ICONST"]" T_SEMI  structdeclare
    | T_CHAR T_ID "["T_ICONST"]" T_SEMI structdeclare |%empty;

functions:function functions

| %empty ;

function: T_FUNCTION T_ID T_LPAREN parameters T_RPAREN body return T_ENDFUNCTION
    {  if (@5.last_line == @6.first_line) {char* lathos ="Body of function must start on
a new line";  yyerror(lathos);  printf("Body of function must start on a new
line");YYABORT;
     }}
|T_FUNCTION T_ID T_LPAREN T_RPAREN body return T_ENDFUNCTION
{  if (@4.last_line == @5.first_line) {char* lathos ="Body of function must start on a
new line" ;yyerror(lathos);  printf("Body of function must start on a new line");YYABORT;
     }};

parameters: parameter T_COMA parameters

|
    parameter
```

```
;

parameter: T_INT Variable

|
T_CHAR Variable

;

Variable: T_ID

|
T_ID "["T_ICONST"]"
;


body: T_VARS Declarations  commands  | %empty  ;

commands:  Assigns commands |If commands|Switch commands|Print commands|while
commands|For commands| %empty;
calc: T_ADDOP | T_MULOP ;

Assigns: parameter T_ASSIGN expr T_SEMI | Variable T_ASSIGN expr  T_SEMI ;

expr:T_ICONST | T_ID | expr calc expr | '-' expr %prec NEG |  expr T_POWER expr |
T_LPAREN expr T_RPAREN  | functionassign ;

functionassign:T_ID T_LPAREN parametersassign T_RPAREN ;

parametersassign:Variable T_COMA Variable;


while: T_WHILE T_LPAREN condition T_RPAREN  bodywhile break T_ENDWHILE ;

bodywhile: commands | break commands ;

break:T_BREAK T_SEMI|%empty;



If: T_IF T_LPAREN condition T_RPAREN T_THEN commands ELSEIF ELSE T_ENDIF  ;

condition: T_ID T_EQUOP T_ID | T_ID T_RELOP T_ID | T_ID T_ANDOP T_ID | T_ID T_OROP T_ID
|T_ICONST;

ELSEIF:T_ELSEIF T_LPAREN condition T_RPAREN commands ELSEIF| %empty;

ELSE:T_ELSE commands |%empty ;

Switch:T_SWITCH T_LPAREN T_ID T_RPAREN CASE DEFAULT T_ENDSWITCH;
```

```
CASE:T_CASE T_LPAREN T_ID T_RPAREN T_COLON commands CASE |T_CASE T_LPAREN T_ICONST
T_RPAREN T_COLON commands CASE | T_CASE T_LPAREN T_CCONST T_RPAREN T_COLON commands CASE
|%empty;

DEFAULT:T_DEFAULT T_COLON commands | %empty;

Print:T_PRINT T_LPAREN  TEXT  VAR T_RPAREN T_SEMI;

TEXT:T_TEXT| %empty;

For:T_FOR T_COUNTER T_ICONST T_TO T_ICONST T_STEP T_ICONST bodywhile break T_ENDFOR ;



VAR: T_LBRACK T_COMA T_ID T_RBRACK |%empty;

Declarations:  Declaration T_SEMI Declarations | Assigns |%empty ;

Declaration:T_INT Variables |  T_CHAR Variables ;

Variables: Variable T_COMA Variables | Variable ;


main: T_STARTMAIN body return T_ENDMAIN ;

return: T_RETURN T_ID T_SEMI |T_RETURN T_ICONST T_SEMI | T_RETURN T_CCONST T_SEMI ;



%%

void yyerror(const char *s) {
    printf(" ----   Error in line: %d\n",yylineno);

    err++;
}

char *readFileContent(const char *const filename)
{
    size_t size;
    FILE  *file;
    char  *data;

    file = fopen(filename, "r");
    if (file == NULL)
    {
        perror("fopen()\n");
        return NULL;
    }

    /* get the file size by seeking to the end and getting the position */
    fseek(file, 0L, SEEK_END);
    size = ftell(file);
    /* reset the file position to the begining. */
    rewind(file);

    /* allocate space to hold the file content */
    data = malloc(1 + size);
    if (data == NULL)
    {
        perror("malloc()\n");
        fclose(file);
        return NULL;
    }
    /* nul terminate the content to make it a valid string */
    data[size] = '\0';
    /* attempt to read all the data */
    if (fread(data, 1, size, file) != size)
    {
        perror("fread()\n");

        free(data);
        fclose(file);

        return NULL;
    }
    fclose(file);
    return data;
}
int main(int argc,char *argv[]){

     ++argv; --argc;
     if (argc > 0) {

        yyin = fopen(argv[0], "r");

        if (!yyin) {
            printf("Error file %s \n",argv[1]);
            exit(1);
        }

    }
    else
        yyin=stdin;

        yyout = fopen ( "output", "w" );

            yyparse();

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

    if (err==0) {
```



```
        char *content;

    content = readFileContent(argv[0]);

    printf("%s\n", content);

    free(content);
        printf(" Systactic ok !\n\n\n");



        }
    else  printf(" Errors: %d !\n\n\n",err);
    return 0;
}
```

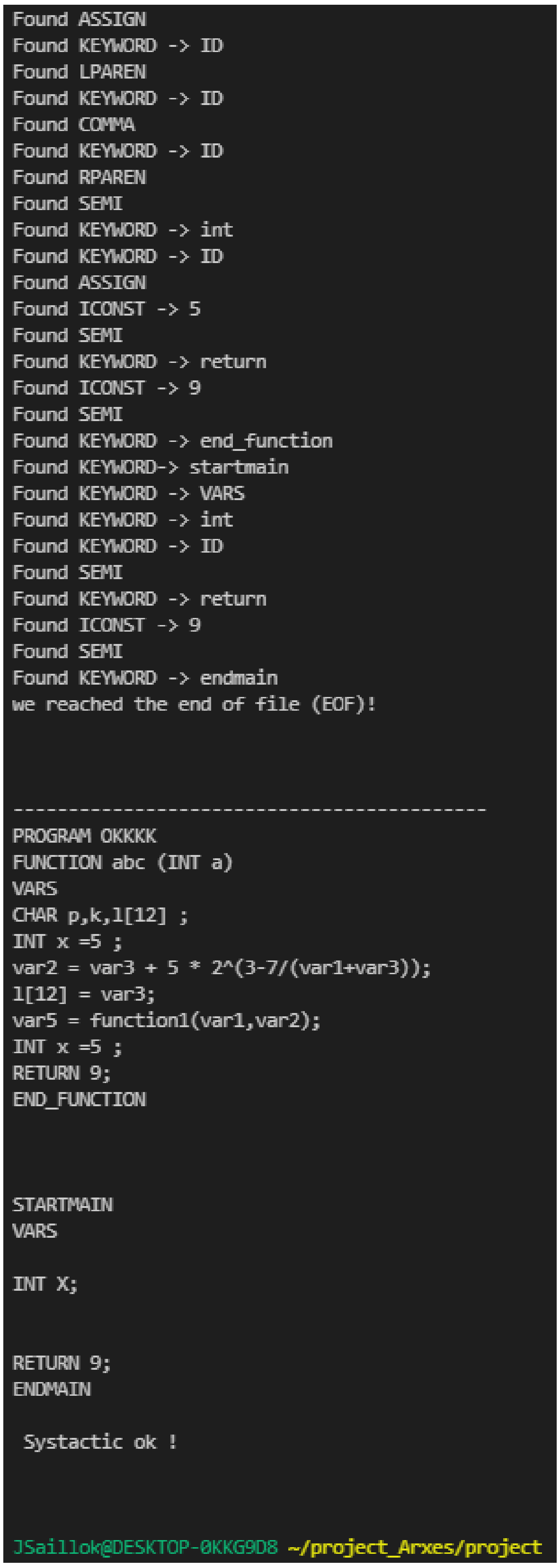
Στην function, για να βεβαιωθούμε ότι υπάρχει αλλαγή γραμμής, χρησιμοποιήσαμε locations για να δούμε αν το επόμενο token ανήκει στην ίδια γραμμή ή την επόμενη (όπου θα έπρεπε να είναι), λόγω εξήγησης που διατυπώσαμε στην flex.

Σύμβολα  παρόμοιας precedence, όπου χρησιμοποιούνται σε ίδιο κανόνα, τα ομαδοποιήσαμε σε ένα token (όπως το + και -).

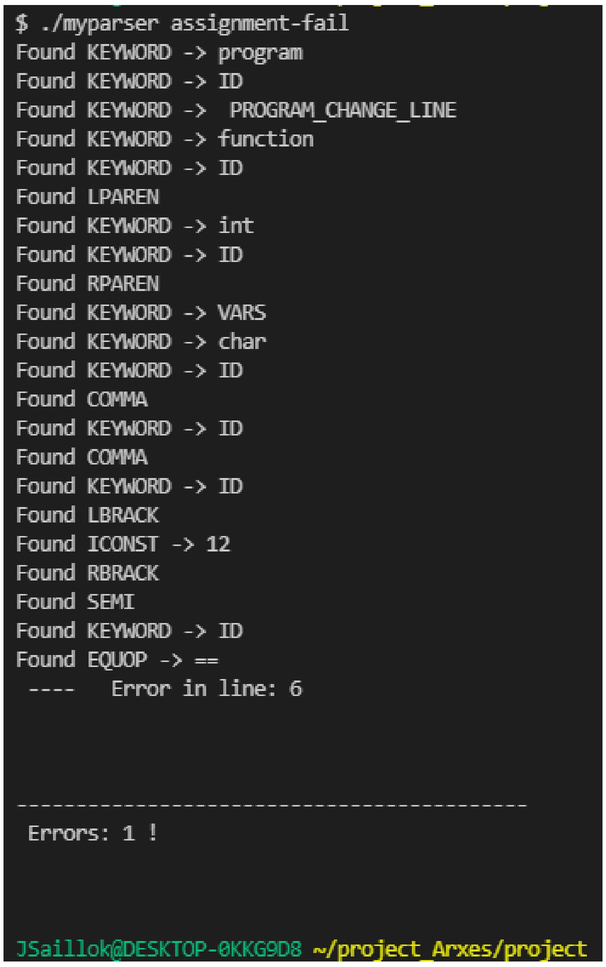
# TESTS-SCREENSHOTS

Ακολουθούν οι διάφορες δοκιμές που κάναμε στο πρόγραμμα:

**Assignment**:

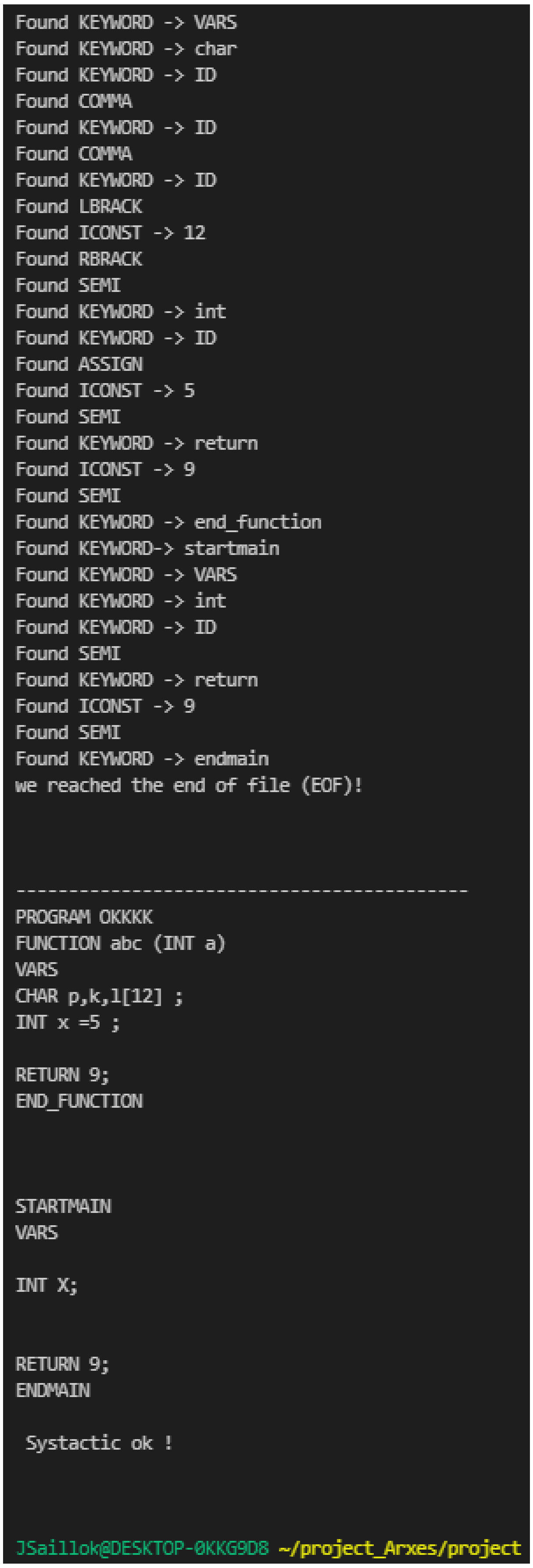


```
Found ASSIGN
Found KEYWORD -> ID
Found LPAREN
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found RPAREN
Found SEMI
Found KEYWORD -> int
Found KEYWORD -> ID
Found ASSIGN
Found ICONST -> 5
Found SEMI
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> end_function
Found KEYWORD-> startmain
Found KEYWORD -> VARS
Found KEYWORD -> int
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> endmain
we reached the end of file (EOF)!



-------------------------------------------
PROGRAM OKKKK
FUNCTION abc (INT a)
VARS
CHAR p,k,l[12] ;
INT x =5 ;
var2 = var3 + 5 * 2^(3-7/(var1+var3));
l[12] = var3;
var5 = function1(var1,var2);
INT x =5 ;
RETURN 9;
END_FUNCTION



STARTMAIN
VARS

INT X;


RETURN 9;
ENDMAIN

 Systactic ok !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

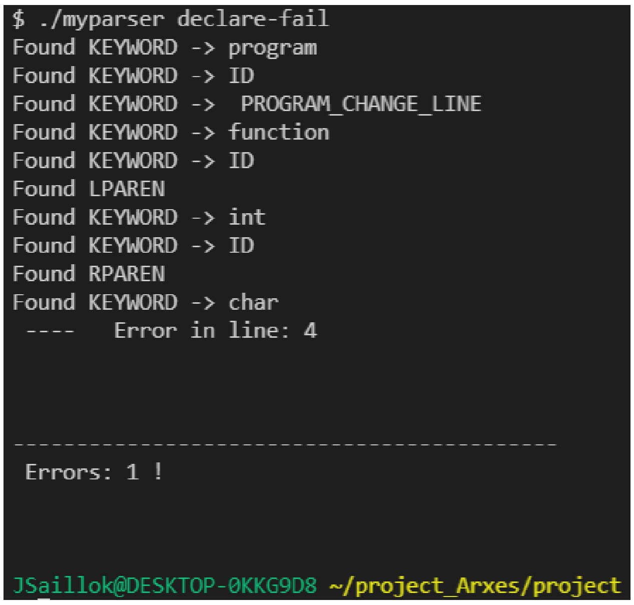


```
$ ./myparser assignment-fail
Found KEYWORD -> program
Found KEYWORD -> ID
Found KEYWORD ->  PROGRAM_CHANGE_LINE
Found KEYWORD -> function
Found KEYWORD -> ID
Found LPAREN
Found KEYWORD -> int
Found KEYWORD -> ID
Found RPAREN
Found KEYWORD -> VARS
Found KEYWORD -> char
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found LBRACK
Found ICONST -> 12
Found RBRACK
Found SEMI
Found KEYWORD -> ID
Found EQUOP -> ==
 ----   Error in line: 6



-------------------------------------------
 Errors: 1 !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

## Declare:

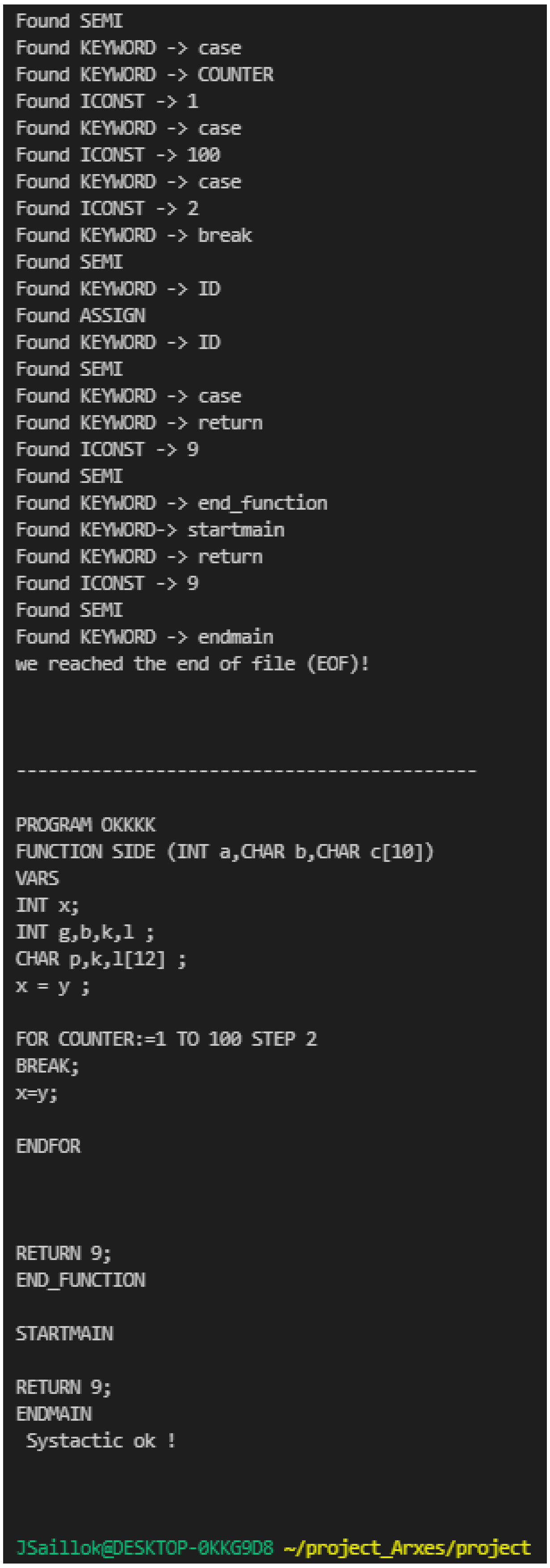


```
Found KEYWORD -> VARS
Found KEYWORD -> char
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found LBRACK
Found ICONST -> 12
Found RBRACK
Found SEMI
Found KEYWORD -> int
Found KEYWORD -> ID
Found ASSIGN
Found ICONST -> 5
Found SEMI
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> end_function
Found KEYWORD-> startmain
Found KEYWORD -> VARS
Found KEYWORD -> int
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> endmain
we reached the end of file (EOF)!



-------------------------------------------
PROGRAM OKKKK
FUNCTION abc (INT a)
VARS
CHAR p,k,l[12] ;
INT x =5 ;

RETURN 9;
END_FUNCTION



STARTMAIN
VARS

INT X;


RETURN 9;
ENDMAIN

 Systactic ok !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

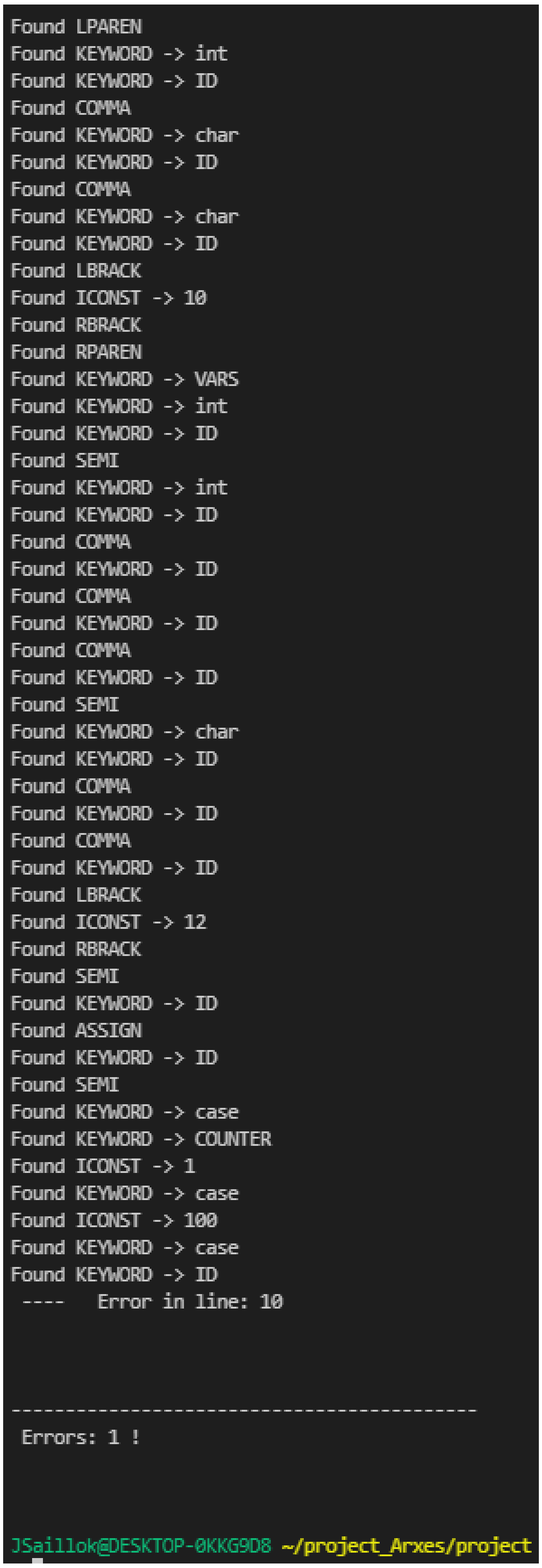


```
$ ./myparser declare-fail
Found KEYWORD -> program
Found KEYWORD -> ID
Found KEYWORD ->  PROGRAM_CHANGE_LINE
Found KEYWORD -> function
Found KEYWORD -> ID
Found LPAREN
Found KEYWORD -> int
Found KEYWORD -> ID
Found RPAREN
Found KEYWORD -> char
 ----   Error in line: 4



---------------------------------------------
 Errors: 1 !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

**For:**

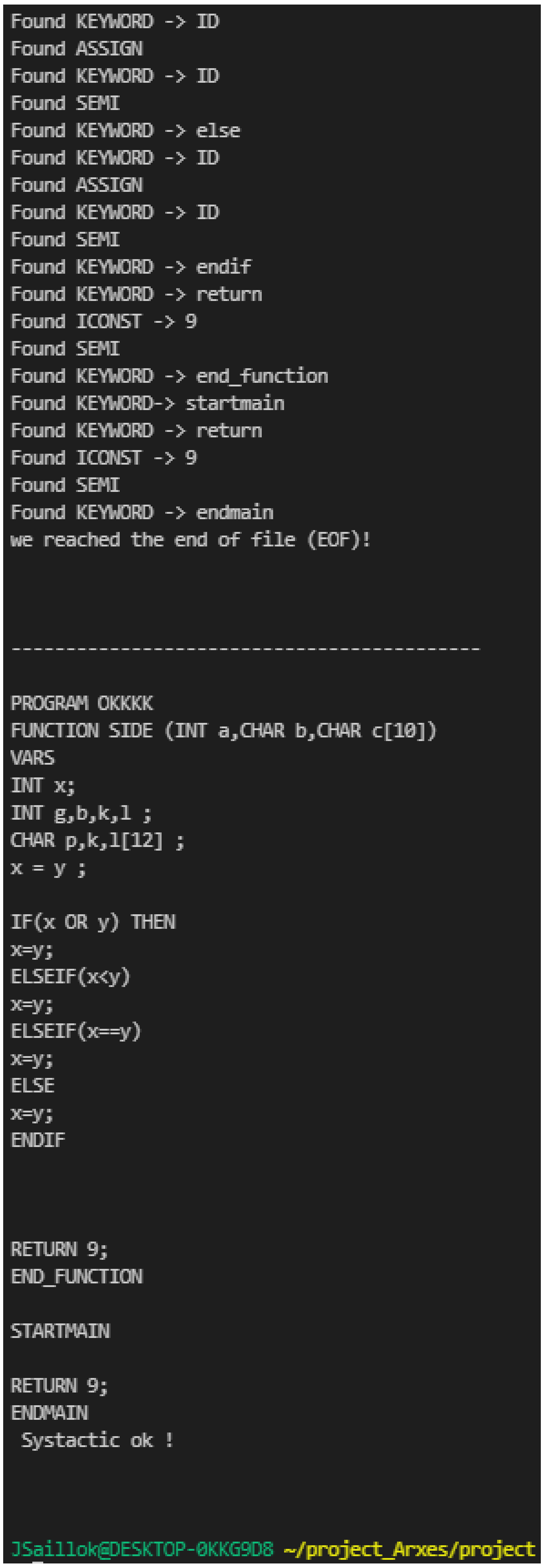


```
Found SEMI
Found KEYWORD -> case
Found KEYWORD -> COUNTER
Found ICONST -> 1
Found KEYWORD -> case
Found ICONST -> 100
Found KEYWORD -> case
Found ICONST -> 2
Found KEYWORD -> break
Found SEMI
Found KEYWORD -> ID
Found ASSIGN
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> case
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> end_function
Found KEYWORD-> startmain
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> endmain
we reached the end of file (EOF)!



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


PROGRAM OKKKK
FUNCTION SIDE (INT a,CHAR b,CHAR c[10])
VARS
INT x;
INT g,b,k,l ;
CHAR p,k,l[12] ;
x = y ;

FOR COUNTER:=1 TO 100 STEP 2
BREAK;
x=y;

ENDFOR



RETURN 9;
END_FUNCTION

STARTMAIN

RETURN 9;
ENDMAIN
 Systactic ok !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

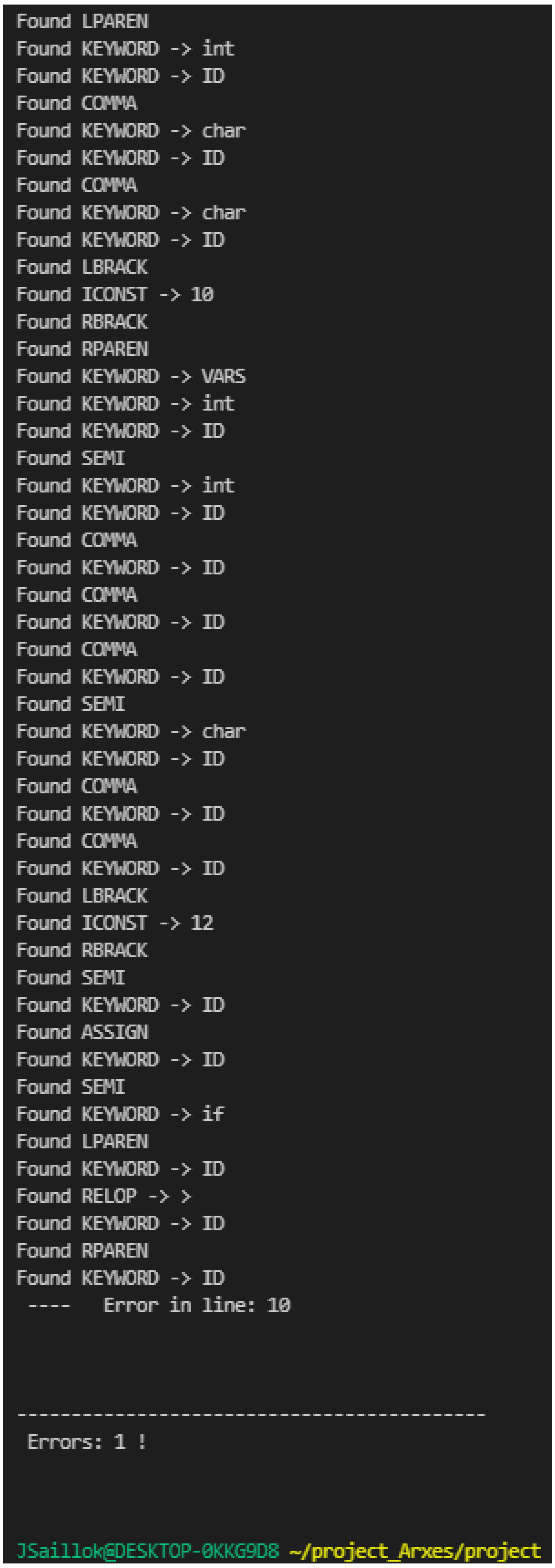


```
Found LPAREN
Found KEYWORD -> int
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> char
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> char
Found KEYWORD -> ID
Found LBRACK
Found ICONST -> 10
Found RBRACK
Found RPAREN
Found KEYWORD -> VARS
Found KEYWORD -> int
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> int
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> char
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found LBRACK
Found ICONST -> 12
Found RBRACK
Found SEMI
Found KEYWORD -> ID
Found ASSIGN
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> case
Found KEYWORD -> COUNTER
Found ICONST -> 1
Found KEYWORD -> case
Found ICONST -> 100
Found KEYWORD -> case
Found KEYWORD -> ID
 ----   Error in line: 10



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

 Errors: 1 !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

**if:**

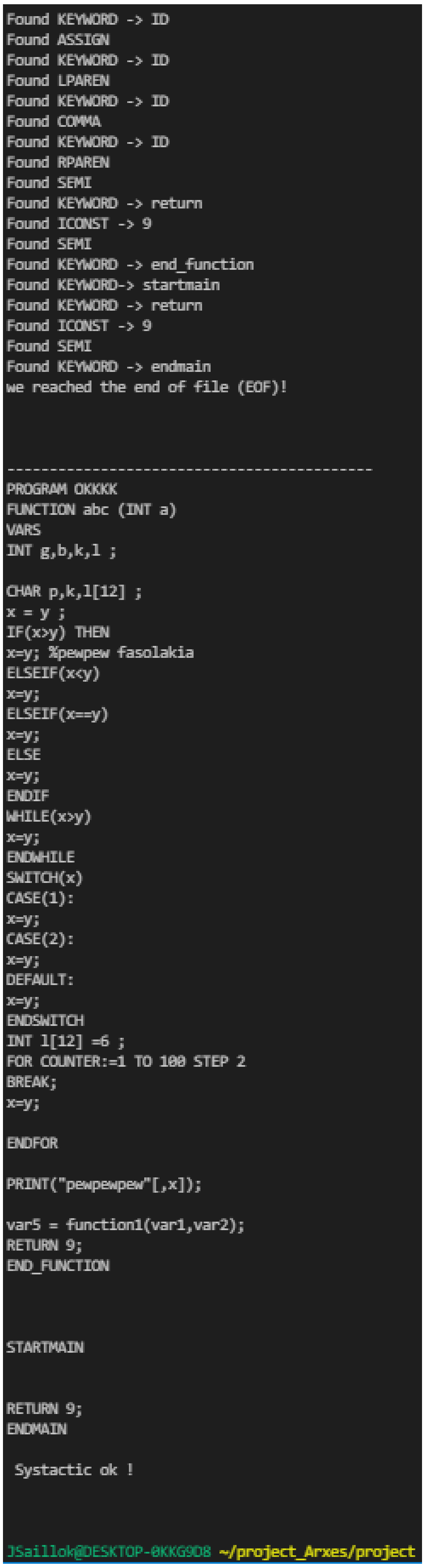


```
Found KEYWORD -> ID
Found ASSIGN
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> else
Found KEYWORD -> ID
Found ASSIGN
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> endif
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> end_function
Found KEYWORD-> startmain
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> endmain
we reached the end of file (EOF)!



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

PROGRAM OKKKK
FUNCTION SIDE (INT a,CHAR b,CHAR c[10])
VARS
INT x;
INT g,b,k,l ;
CHAR p,k,l[12] ;
x = y ;

IF(x OR y) THEN
x=y;
ELSEIF(x<y)
x=y;
ELSEIF(x==y)
x=y;
ELSE
x=y;
ENDIF



RETURN 9;
END_FUNCTION

STARTMAIN

RETURN 9;
ENDMAIN
 Systactic ok !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

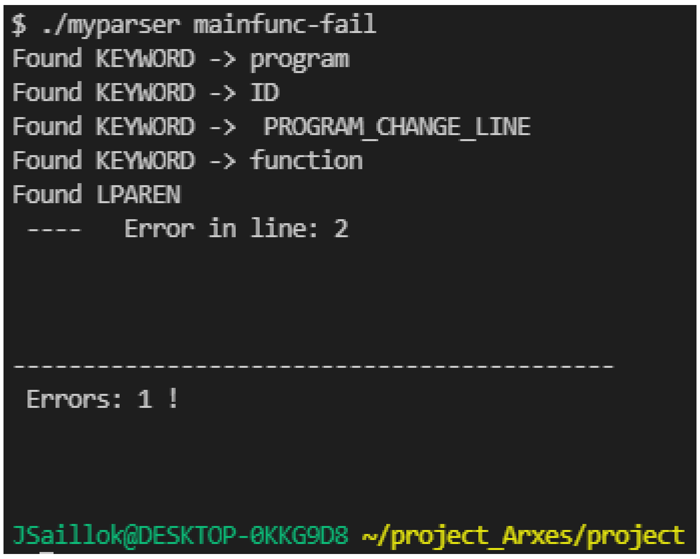


```
Found LPAREN
Found KEYWORD -> int
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> char
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> char
Found KEYWORD -> ID
Found LBRACK
Found ICONST -> 10
Found RBRACK
Found RPAREN
Found KEYWORD -> VARS
Found KEYWORD -> int
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> int
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> char
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found LBRACK
Found ICONST -> 12
Found RBRACK
Found SEMI
Found KEYWORD -> ID
Found ASSIGN
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> if
Found LPAREN
Found KEYWORD -> ID
Found RELOP -> >
Found KEYWORD -> ID
Found RPAREN
Found KEYWORD -> ID
 ----   Error in line: 10



-------------------------------------------
 Errors: 1 !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

**main function:**

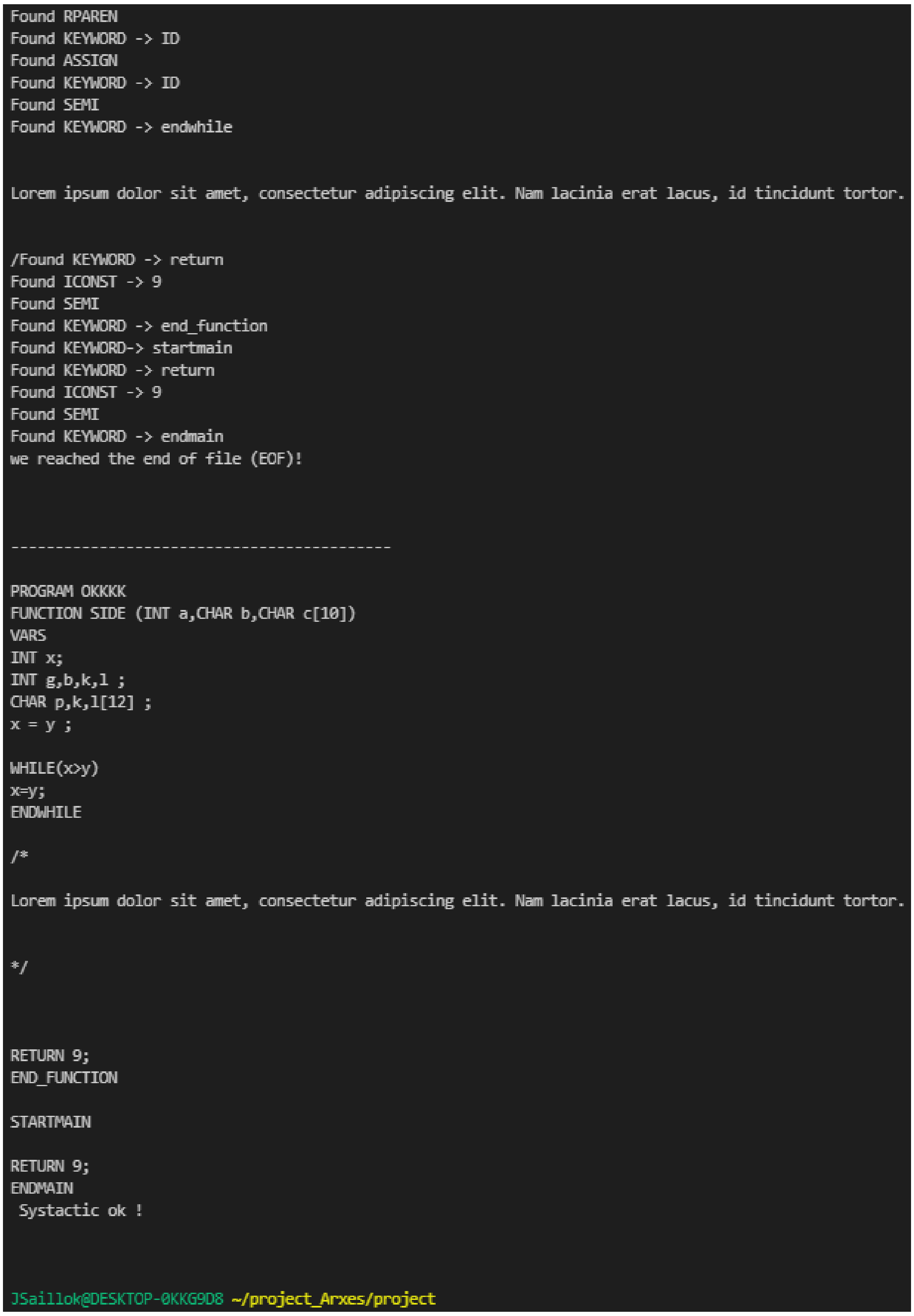


```
Found KEYWORD -> ID
Found ASSIGN
Found KEYWORD -> ID
Found LPAREN
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found RPAREN
Found SEMI
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> end_function
Found KEYWORD-> startmain
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> endmain
we reached the end of file (EOF)!



-------------------------------------------
PROGRAM OKKKK
FUNCTION abc (INT a)
VARS
INT g,b,k,l ;

CHAR p,k,l[12] ;
x = y ;
IF(x>y) THEN
x=y; %pewpew fasolakia
ELSEIF(x<y)
x=y;
ELSEIF(x==y)
x=y;
ELSE
x=y;
ENDIF
WHILE(x>y)
x=y;
ENDWHILE
SWITCH(x)
CASE(1):
x=y;
CASE(2):
x=y;
DEFAULT:
x=y;
ENDSWITCH
INT l[12] =6 ;
FOR COUNTER:=1 TO 100 STEP 2
BREAK;
x=y;

ENDFOR

PRINT("pewpewpew"[,x]);

var5 = function1(var1,var2);
RETURN 9;
END_FUNCTION



STARTMAIN


RETURN 9;
ENDMAIN

 Systactic ok !


JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

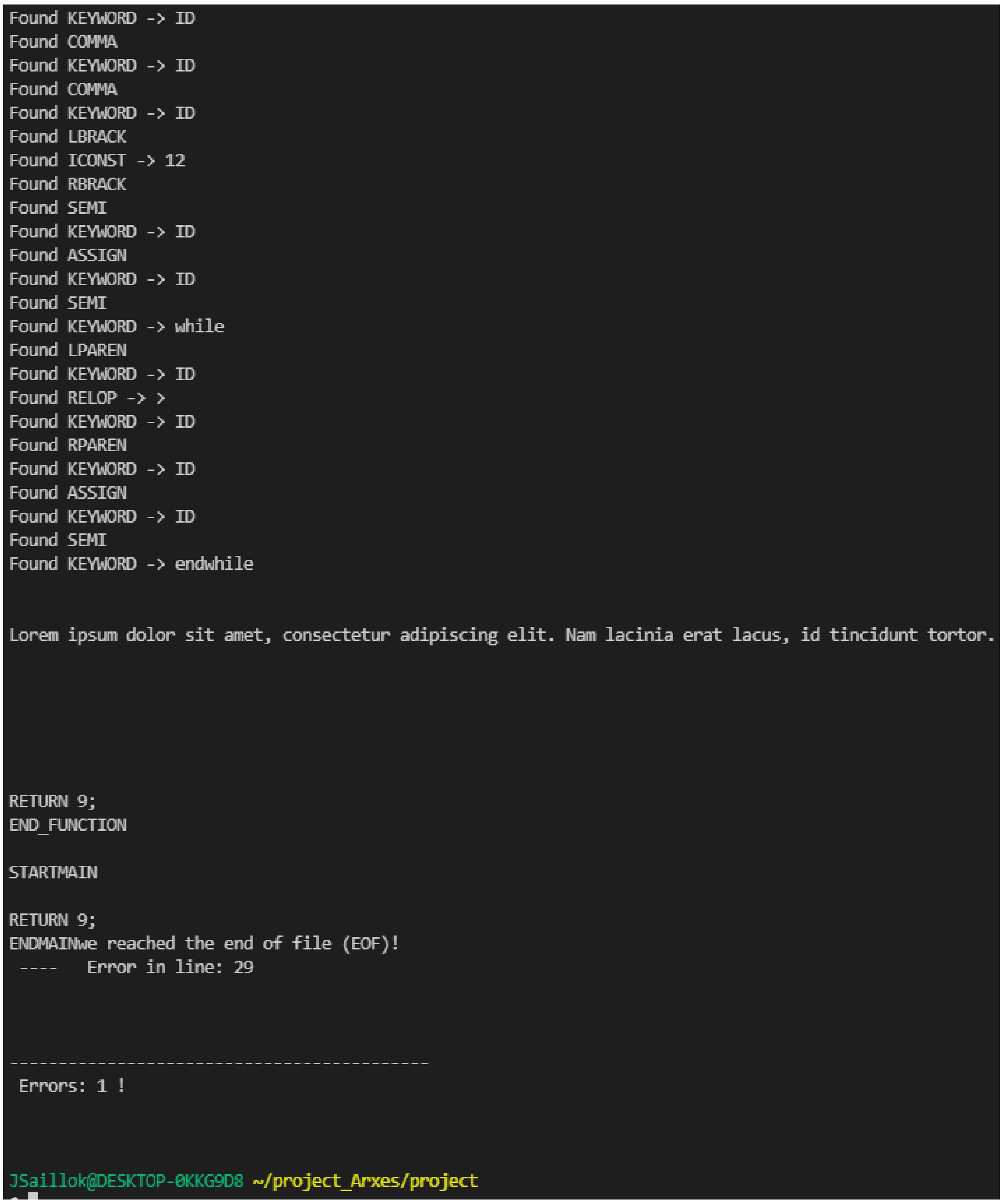


```
$ ./myparser mainfunc-fail
Found KEYWORD -> program
Found KEYWORD -> ID
Found KEYWORD ->  PROGRAM_CHANGE_LINE
Found KEYWORD -> function
Found LPAREN
 ----    Error in line: 2



---------------------------------------------
 Errors: 1 !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

**multiline comment:**

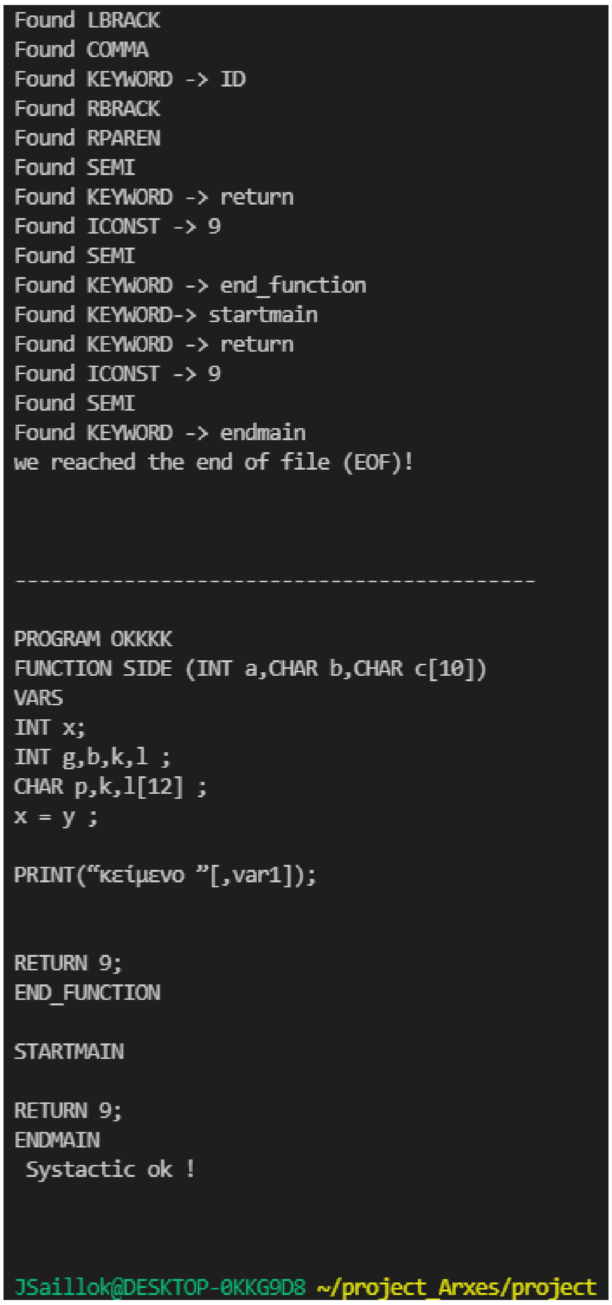


```
Found RPAREN
Found KEYWORD -> ID
Found ASSIGN
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> endwhile


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam lacinia erat lacus, id tincidunt tortor.


/Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> end_function
Found KEYWORD-> startmain
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> endmain
we reached the end of file (EOF)!



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

PROGRAM OKKKK
FUNCTION SIDE (INT a,CHAR b,CHAR c[10])
VARS
INT x;
INT g,b,k,l ;
CHAR p,k,l[12] ;
x = y ;

WHILE(x>y)
x=y;
ENDWHILE

/*

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam lacinia erat lacus, id tincidunt tortor.


*/



RETURN 9;
END_FUNCTION

STARTMAIN

RETURN 9;
ENDMAIN
 Systactic ok !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

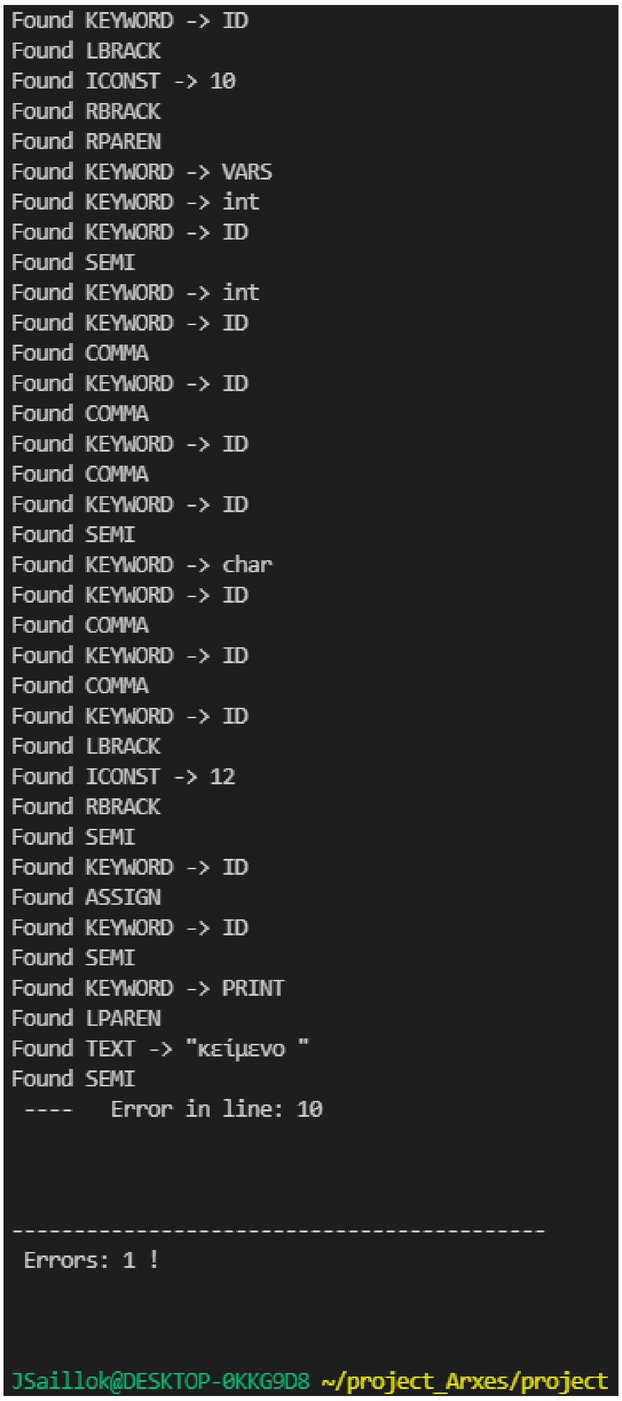


```
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found LBRACK
Found ICONST -> 12
Found RBRACK
Found SEMI
Found KEYWORD -> ID
Found ASSIGN
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> while
Found LPAREN
Found KEYWORD -> ID
Found RELOP -> >
Found KEYWORD -> ID
Found RPAREN
Found KEYWORD -> ID
Found ASSIGN
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> endwhile


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam lacinia erat lacus, id tincidunt tortor.






RETURN 9;
END_FUNCTION

STARTMAIN

RETURN 9;
ENDMAINwe reached the end of file (EOF)!
 ----   Error in line: 29




-------------------------------------------
 Errors: 1 !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

**Print:**

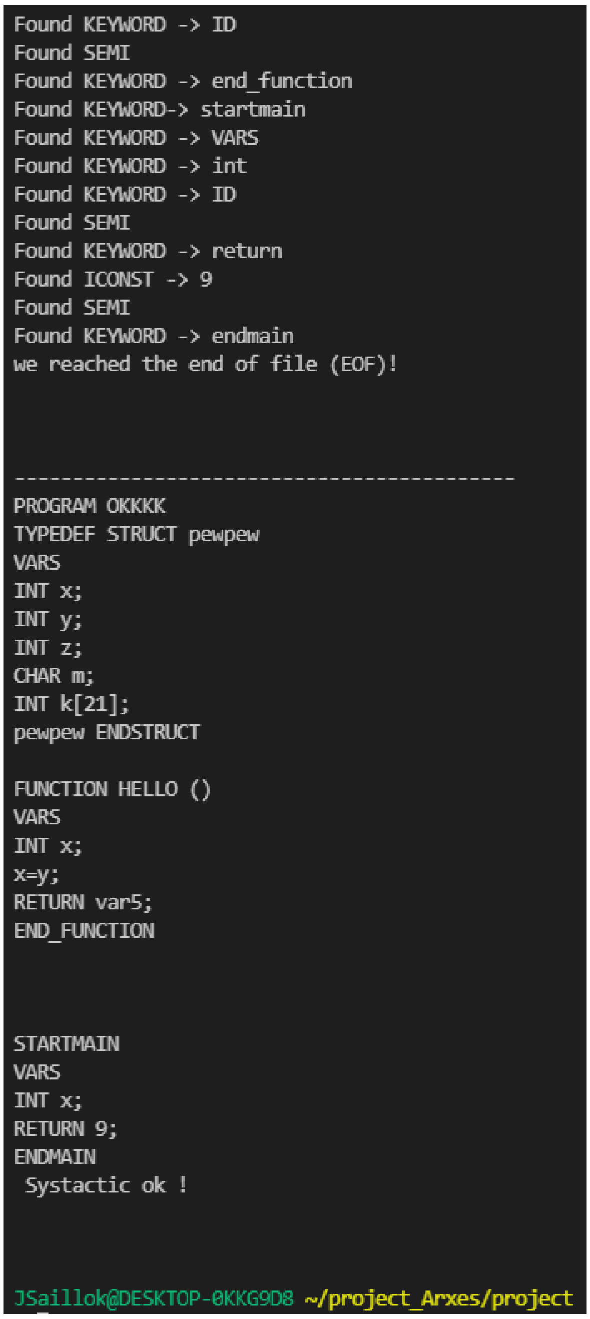


```
Found LBRACK
Found COMMA
Found KEYWORD -> ID
Found RBRACK
Found RPAREN
Found SEMI
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> end_function
Found KEYWORD-> startmain
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> endmain
we reached the end of file (EOF)!



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

PROGRAM OKKKK
FUNCTION SIDE (INT a,CHAR b,CHAR c[10])
VARS
INT x;
INT g,b,k,l ;
CHAR p,k,l[12] ;
x = y ;

PRINT(“κείμενο ”[,var1]);


RETURN 9;
END_FUNCTION

STARTMAIN

RETURN 9;
ENDMAIN
 Systactic ok !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

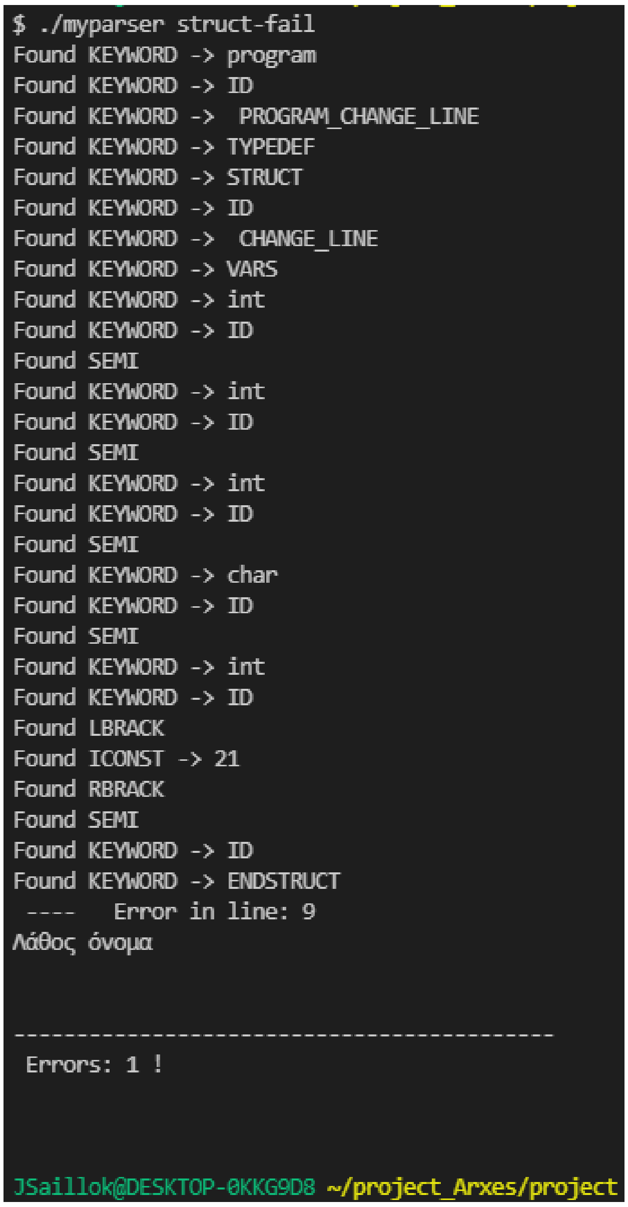


```
Found KEYWORD -> ID
Found LBRACK
Found ICONST -> 10
Found RBRACK
Found RPAREN
Found KEYWORD -> VARS
Found KEYWORD -> int
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> int
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> char
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found LBRACK
Found ICONST -> 12
Found RBRACK
Found SEMI
Found KEYWORD -> ID
Found ASSIGN
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> PRINT
Found LPAREN
Found TEXT -> "κείμενο "
Found SEMI
 ----   Error in line: 10



-------------------------------------------
 Errors: 1 !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

**Struct:**

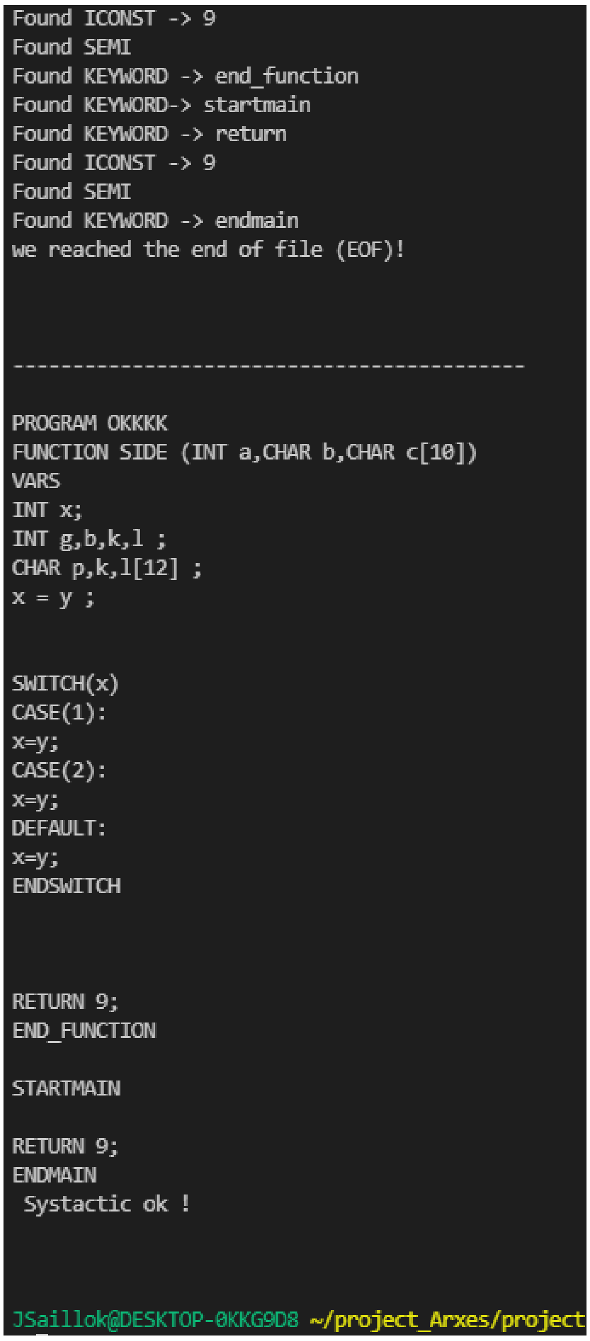


```
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> end_function
Found KEYWORD-> startmain
Found KEYWORD -> VARS
Found KEYWORD -> int
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> endmain
we reached the end of file (EOF)!



-------------------------------------------
PROGRAM OKKKK
TYPEDEF STRUCT pewpew
VARS
INT x;
INT y;
INT z;
CHAR m;
INT k[21];
pewpew ENDSTRUCT

FUNCTION HELLO ()
VARS
INT x;
x=y;
RETURN var5;
END_FUNCTION



STARTMAIN
VARS
INT x;
RETURN 9;
ENDMAIN
 Systactic ok !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

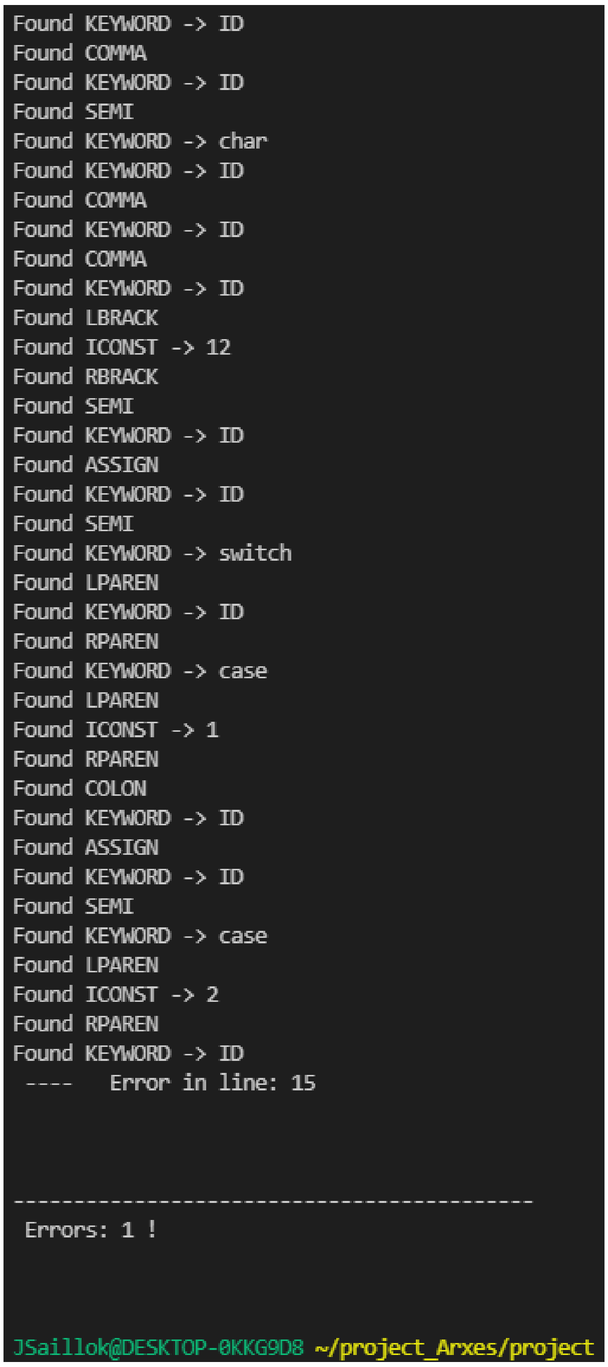


```
$ ./myparser struct-fail
Found KEYWORD -> program
Found KEYWORD -> ID
Found KEYWORD ->  PROGRAM_CHANGE_LINE
Found KEYWORD -> TYPEDEF
Found KEYWORD -> STRUCT
Found KEYWORD -> ID
Found KEYWORD ->  CHANGE_LINE
Found KEYWORD -> VARS
Found KEYWORD -> int
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> int
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> int
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> char
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> int
Found KEYWORD -> ID
Found LBRACK
Found ICONST -> 21
Found RBRACK
Found SEMI
Found KEYWORD -> ID
Found KEYWORD -> ENDSTRUCT
 ----   Error in line: 9
Λάθος όνομα


-------------------------------------------
 Errors: 1 !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

**Switch:**

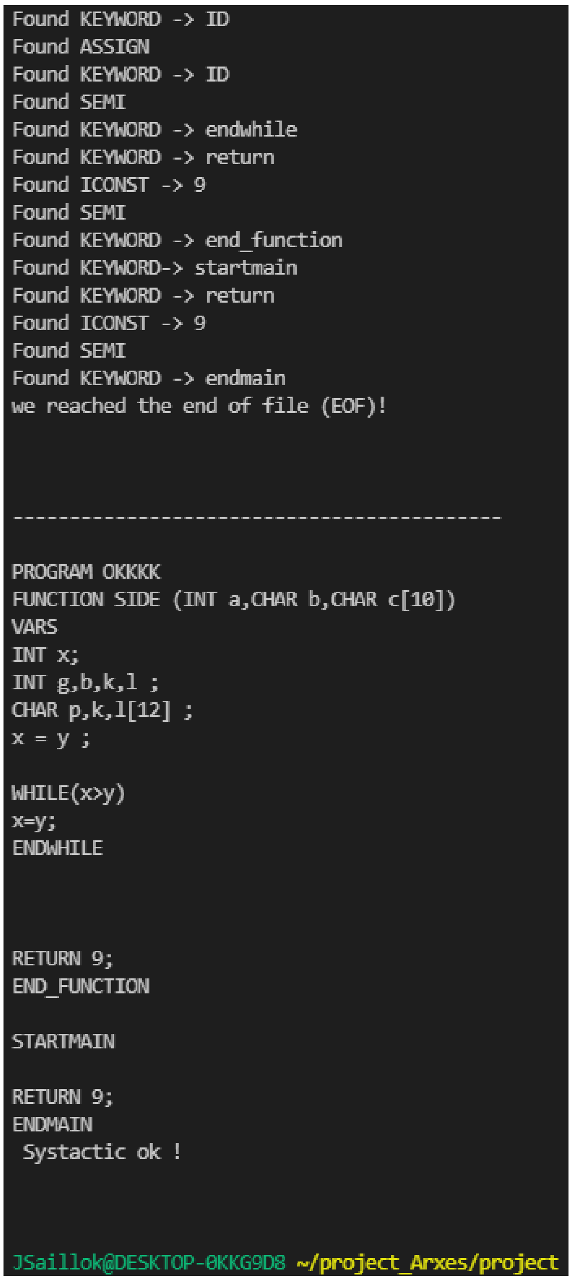


```
Found ICONST -> 9
Found SEMI
Found KEYWORD -> end_function
Found KEYWORD-> startmain
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> endmain
we reached the end of file (EOF)!



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

PROGRAM OKKKK
FUNCTION SIDE (INT a,CHAR b,CHAR c[10])
VARS
INT x;
INT g,b,k,l ;
CHAR p,k,l[12] ;
x = y ;


SWITCH(x)
CASE(1):
x=y;
CASE(2):
x=y;
DEFAULT:
x=y;
ENDSWITCH



RETURN 9;
END_FUNCTION

STARTMAIN

RETURN 9;
ENDMAIN
 Systactic ok !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

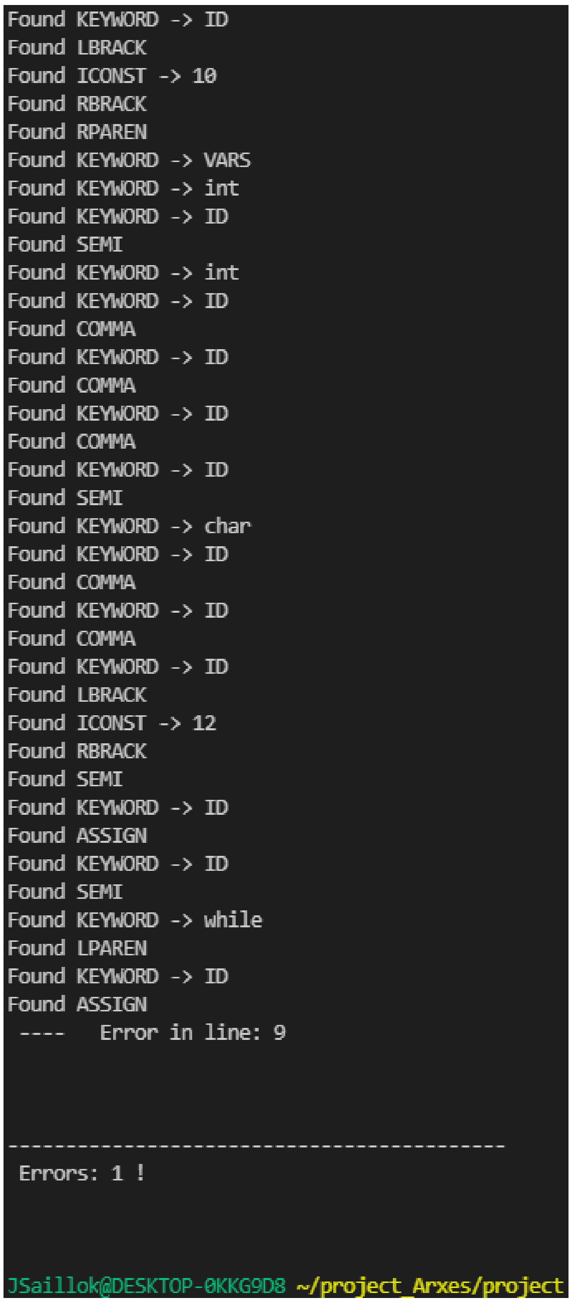


```
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> char
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found LBRACK
Found ICONST -> 12
Found RBRACK
Found SEMI
Found KEYWORD -> ID
Found ASSIGN
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> switch
Found LPAREN
Found KEYWORD -> ID
Found RPAREN
Found KEYWORD -> case
Found LPAREN
Found ICONST -> 1
Found RPAREN
Found COLON
Found KEYWORD -> ID
Found ASSIGN
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> case
Found LPAREN
Found ICONST -> 2
Found RPAREN
Found KEYWORD -> ID
 ----    Error in line: 15



-------------------------------------------
 Errors: 1 !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

## While:

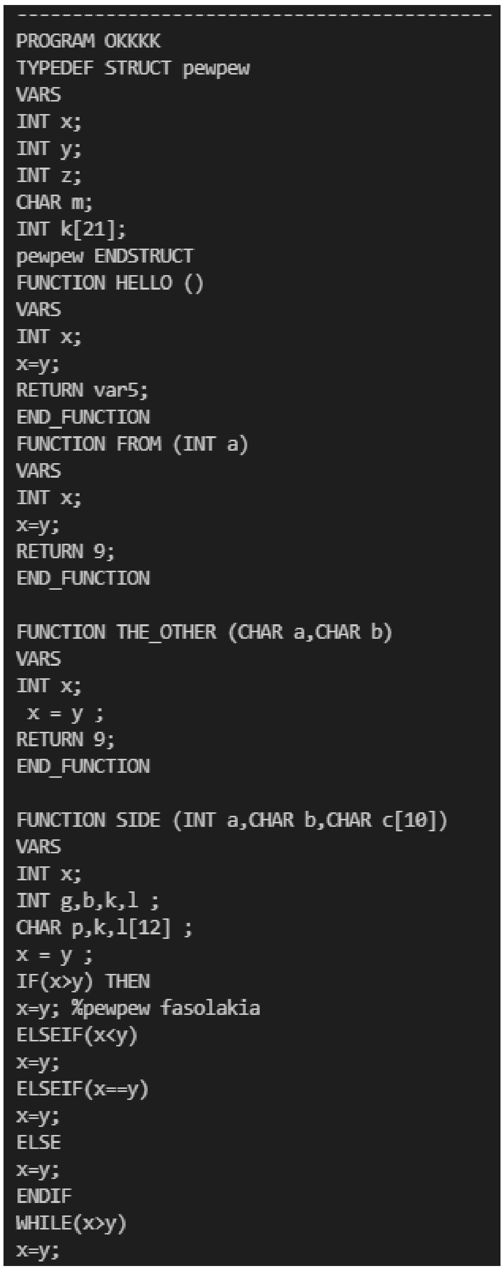


```
Found KEYWORD -> ID
Found ASSIGN
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> endwhile
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> end_function
Found KEYWORD-> startmain
Found KEYWORD -> return
Found ICONST -> 9
Found SEMI
Found KEYWORD -> endmain
we reached the end of file (EOF)!



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


PROGRAM OKKKK
FUNCTION SIDE (INT a,CHAR b,CHAR c[10])
VARS
INT x;
INT g,b,k,l ;
CHAR p,k,l[12] ;
x = y ;

WHILE(x>y)
x=y;
ENDWHILE



RETURN 9;
END_FUNCTION

STARTMAIN

RETURN 9;
ENDMAIN
 Systactic ok !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

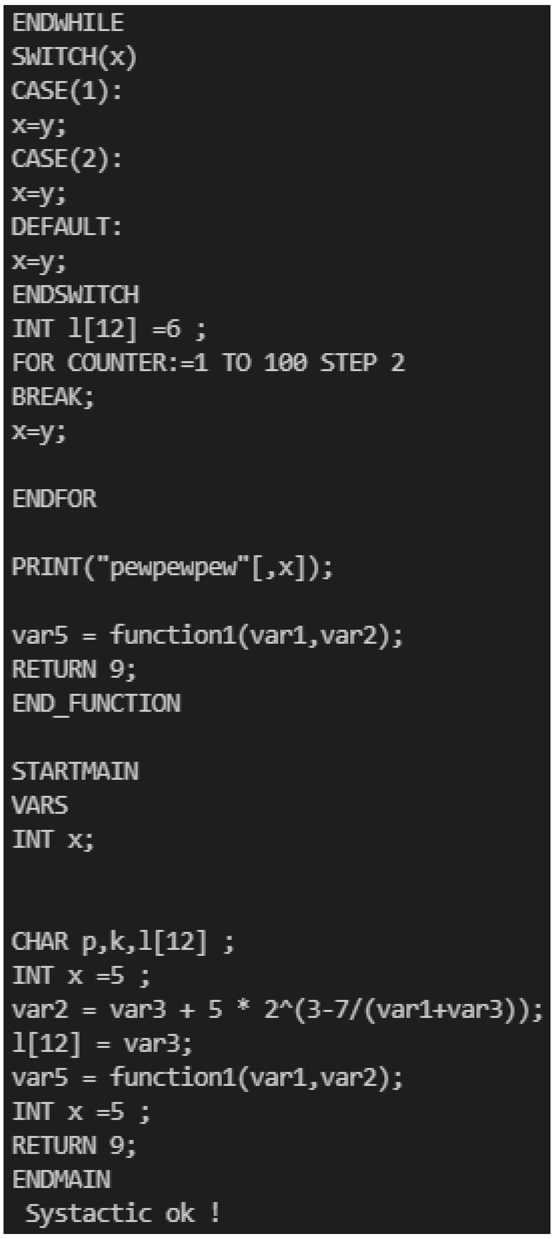


```
Found KEYWORD -> ID
Found LBRACK
Found ICONST -> 10
Found RBRACK
Found RPAREN
Found KEYWORD -> VARS
Found KEYWORD -> int
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> int
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> char
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found COMMA
Found KEYWORD -> ID
Found LBRACK
Found ICONST -> 12
Found RBRACK
Found SEMI
Found KEYWORD -> ID
Found ASSIGN
Found KEYWORD -> ID
Found SEMI
Found KEYWORD -> while
Found LPAREN
Found KEYWORD -> ID
Found ASSIGN
 ----   Error in line: 9



------------------------------------------
 Errors: 1 !



JSaillok@DESKTOP-0KKG9D8 ~/project_Arxes/project
```

**Πλήρης δοκιμή του Προγράμματος:**



```
------------------------------------------
PROGRAM OKKKK
TYPEDEF STRUCT pewpew
VARS
INT x;
INT y;
INT z;
CHAR m;
INT k[21];
pewpew ENDSTRUCT
FUNCTION HELLO ()
VARS
INT x;
x=y;
RETURN var5;
END_FUNCTION
FUNCTION FROM (INT a)
VARS
INT x;
x=y;
RETURN 9;
END_FUNCTION

FUNCTION THE_OTHER (CHAR a,CHAR b)
VARS
INT x;
 x = y ;
RETURN 9;
END_FUNCTION

FUNCTION SIDE (INT a,CHAR b,CHAR c[10])
VARS
INT x;
INT g,b,k,l ;
CHAR p,k,l[12] ;
x = y ;
IF(x>y) THEN
x=y; %pewpew fasolakia
ELSEIF(x<y)
x=y;
ELSEIF(x==y)
x=y;
ELSE
x=y;
ENDIF
WHILE(x>y)
x=y;
```



```
ENDWHILE
SWITCH(x)
CASE(1):
x=y;
CASE(2):
x=y;
DEFAULT:
x=y;
ENDSWITCH
INT l[12] =6 ;
FOR COUNTER:=1 TO 100 STEP 2
BREAK;
x=y;

ENDFOR

PRINT("pewpewpew"[,x]);

var5 = function1(var1,var2);
RETURN 9;
END_FUNCTION

STARTMAIN
VARS
INT x;


CHAR p,k,l[12] ;
INT x =5 ;
var2 = var3 + 5 * 2^(3-7/(var1+var3));
l[12] = var3;
var5 = function1(var1,var2);
INT x =5 ;
RETURN 9;
ENDMAIN
 Systactic ok !
```