Αρχές Γλωσσών Προγραμματισμού και Μεταφραστών

Αναφορά Εργαστηριακής Άσκησης 2020-2021

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΗΛΕΚΤΡΟΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

Φοιτητές: Άγκο Μπεσιάνα 1059662

Σπεντζάρης Παναγιώτης 1071110

Μπότσας Γεώργιος 1070929

Νάνος Γεώργιος 1059547

Καθηγητές: Ιωάννης Γαροφαλάκης

Σπυρίδων Σιούτας

Πίνακας περιεχομένων

1. Εισαγωγή

Σε αυτήν την εργαστηριακή άσκηση ασχοληθήκαμε με την περιγραφή μιας γλώσσας (ψευδογλώσσα) η οποία ακολουθεί την λογική της C. Μας ζητήθηκε να υλοποιήσουμε δύο από τα τμήματα ενός μεταγλωττιστή, έναν συντακτικό και έναν λεκτικό αναλυτή με τη χρήση των εργαλείων Bison και Flex.

Ο Λεκτικός αναλυτής (Flex) διαβάζει χαρακτήρες από την είσοδο μας και τους ταιριάζει σε patterns, παράγοντας lexemes, τα οποία αντιστοιχίζονται σε που αποτελούν την είσοδο του συντακτικού αναλυτή.

Ο Συντακτικός αναλυτής (Bison) δέχεται μια γραμματική χωρίς συμφραζόμενα και παράγει έναν συντακτικό αναλυτή σε C. Η παραγόμενη συνάρτηση αναγνωρίζει τις συμβολοσειρές εισόδου, κατασκευάζει το συντακτικό δέντρο και εκτελεί τις ενέργειες που περιγράφονται στο πρόγραμμα.

1.1 Σχόλια και περιγραφή BNF

Το πρόγραμμα μας ξεκινά με την δεσμευμένη λέξη **PROGRAM** ακολουθούμενη από το όνομα του προγράμματος και υποχρεωτική αλλαγή γραμμής .

Ακολουθεί ο προαιρετικός ορισμός συναρτήσεων. Οι συναρτήσεις ξεκινούν με τη δεσμευμένη λέξη **FUNCTION**, από το όνομα της συνάρτησης και στη συνέχεια, σε παρένθεση, τις παραμέτρους χωρισμένες με κόμμα. Στη συνέχεια έχουμε το σώμα της συνάρτησης με τη προεραιτική δήλωση μεταβλητών. Η δήλωση πραγματοποιείται με τη λεξη **VARS**. Πρώτα ορίσαμε τον τύπο δεδομένων των μεταβλητών ακολουθούμενος από λίστα με τα ονόματα των μεταβλητών χωρισμένων με “,” και στο τέλος υπάρχει “;”. Στο τέλος της συνάρτησης υπάρχει η δεσμευμένη λέξη **END_FUNCTION** , αλλά πιο πριν υποχρεωτικά επιστρέφεται μια τιμή με τη λέξη **RETURN**.

Έπειτα έχουμε το κύριο μέρος του προγράμματος που περικλείεται από τις λέξεις **STARTMAIN** και **ENDMAIN**, το οποίο περιλαμβάνει την προαιρετική δήλωση μεταβλητών τις εντολές του προγράμματος με οποιαδήποτε σειρά.

Οι εντολές προγράμματος διακρίνονται σε :

- Εντολές ανάθεσης

  Είναι της μορφής **<μεταβλητή>=<έκφραση>;** και περιέχει οποιαδήποτε αριθμητική παράσταση η οποία περιλαμβάνει τις πράξεις **+,=,*, ^ ,/.**

- Εντολές βρόχου

  Διαθέτουμε δύο διαφορετικά είδη εντολών βρόχου.

  Αρχικά, έχουμε τις while εντολές οι οποίες ξεκινούν με τη λέξη **WHILE** ακολουθούμενη από τη **συνθήκη** και τις εντολές προγράμματος και τελειώνει με τη λέξη **ENDWHILE** .

  Το δεύτερο είδος είναι οι for εντολές οι οποίες είναι της μορφής :
  FOR counter:=… **TO** … **STEP** … εντολές προγράμματος ..
  Και τελειώνουν με τη δεσμευμένη λέξη **ENDFOR**.

  Η συνθήκη περιλαμβάνει οποιαδήποτε έκφραση η οποια περιέχει είτε συγκριτικούς (π.χ. **>**, **<**,**==**,**!=**) είτε λογικούς τελεστές (π.χ. **AND**,**OR**).

- Εντολές ελέγχου

  Έχουμε δύο είδη εντολών ελέγχου.

  Τις if εντολές που έχουν την μορφή :
  IF (συνθήκη) **THEN**
  … εντολές προγράμματος …
  ELSEIF
  … εντολές προγράμματος …
  …
  ELSE
  … εντολές προγράμματος …
  ENDIF

  Όπου η εμφάνιση των ELSEIF και ELSE είναι προαιρετική.

Τις switch εντολές με την ακόλουθη μορφή :
SWITCH (έκφραση)
CASE (έκφραση**):**
… εντολές προγράμματος …
…
DEFAULT:
… εντολές προγράμματος …
ENDSWITCH

Όπου η εντολή default είναι προαιρετική.

- Εντολές εκτύπωσης

  Οι εντολές εκτύπωσης ξεκινούν με τη λέξη **PRINT** και ακολουθεί (“κείμενο”,[var1]);

- Εντολή τερματισμού βρόχου

  Η εντολή τερματισμού βρόχου αποτελείται από τη δεσμευμένη λέξη **BREAK**

Μετά τον προαιρετικό ορισμό συναρτήσεων μπορούμε να έχουμε τη δήλωση τύπου δεδομένων χρήστη, η οποία ορίζεται με τη δεσμευμένη λέξη **STRUCT** ακολουθούμενη από το όνομα του τύπου. Έπειτα μετά από αλλαγή γραμμής τις δηλώσεις μεταβλητών με τη λέξη **VARS** και τελειώνει με τη δεσμευμένη λέξη **ENDSTRUCT**.

Επίσης, πριν από τη δεσμευμένη λέξη STRUCT (προαιρετικά) μπορεί να εμφανίζεται η δεσμευμένη λέξη **TYPEDEF** (π.χ. TYPEDEF STRUCT <όνομα τύπου>). Στη συνέχεια, ακολουθούν οι δηλώσεις μεταβλητών. Το τέλος του ορισμού σε αυτή την περίπτωση ορίζεται ως **ENDSTRUCT <όνομα τύπου>**.

2. Ερωτήματα

2.1.α BNF Συντακτικός ορισμός ψευδογλώσσας

```
PROGRAM: T_PROGRAM name {printf("\n");} body_program ;

name: T_charident ;

Struct : T_STRUCT name {printf("\n");} variable_declaration T_ENDSTRUCT
  |T_TYPEDEF T_STRUCT name {printf("\n");} variable_declaration {printf("\n");} name
T_ENDSTRUCT
  ;

body_program : Struct Functions  Main_function
   | Struct Main_function
   | Main_function
   ;

Functions : T_FUNCTION name T_open P1 T_close body_Function T_return T_charident
{printf("\n");} T_end ;

P1 : %empty
   | V1 P1
   ;

V1 : type_variable d
   | type_variable d T_komma V1
   ;
```

```
d  : name
   | table
   ;

table :T_pin ;

type_variable : T_int
              | T_float
              | T_char
              ;

body_Function : variable_declaration program_commands
           | variable_declaration
           | %empty
           ;

variable_declaration : T_VARS format  T_semicolon ; // format = tropos dilosis

format : type_variable P2 ;

P2 : V2
   | V2 P2
   ;
V2 : d
   | d T_komma | T_komma d G
   ;

G  : %empty
   ;
```

```
program_commands : assign_commands extra_commands
                 | assign_commands
                                        | assign_commands program_commands
                 | extra_commands
                 | %empty
                 ;


assign_commands: variable T_ASSIGN expression T_semicolon ;


expression:  oros
                     | function
                     ;

function: name T_open orisma T_close  ;


orisma: variable
     | variable T_komma orisma
           ;


oros: A
           | A telestes A oros
           | A telestes parenthesi oros
           ;


parenthesi: T_open A T_close
         | T_open A T_close parenthesi ;
```

```
variable : d ;

A: variable
  | variable telestes A
  | number
  | number telestes A
  ;

 telestes: T_adop
        | T_mulop
                    ;

number : T_intident
       | T_floatident
       ;

Main_function: T_STARTMAIN variable_declaration program_commands T_ENDMAIN
            |  T_STARTMAIN program_commands T_ENDMAIN
            ;

extra_commands : H ;


H : Q
  |Q H
  ;

Q : loop_commands control_commands print_command
             | loop_commands print_command
```

```
        | control_commands print_command
        | loop_commands  control_commands
        | print_command
        | loop_commands
        | control_commands
        ;

loop_commands : For_command
         | While_command
         ;

For_command : T_FOR counter T_colon T_ASSIGN number T_TO number T_STEP number
{printf("\n");} program_commands L T_ENDFOR ;

//type :T_intident
    | T_charident ;//

counter : variable ;

While_command : T_WHILE T_open condition T_close {printf("\n");} program_commands L
T_ENDWHILE ;

comparison_operator: T_relop
 | T_equop
 ;

logic_operators: T_ANDOP
 | T_OROP
 ;
```

```
L: %empty
 | break_command L
 ;

break_command : T_BREAK T_semicolon ;

print_command : T_PRINT T_open T_string X T_close T_semicolon ;

X : %empty
  | F X  ;

F :   T_komma d  ;

control_commands: If_command
              | Switch_command
              | If_command Switch_command
             ;

If_command : T_IF T_open condition T_close T_THEN {printf("\n");} program_commands
elseif else L T_ENDIF ;

condition : P
         | T
                    ;

 P: A comparison_operator A ;

 T: %empty
  | J T
  ;
```

```
J: P logic_operators P
 | logic_operators P
 ;

elseif : %empty
     | T_ELSEIF  program_commands elseif
     ;

else : %empty
     | T_ELSE program_commands
     ;

Switch_command : T_SWITCH  T_open expression T_close {printf("\n");} case
T_ENDSWITCH ;

case : M L
    | M L case
    | M L default L
    ;

M : T_CASE T_open expression T_close T_colon {printf("\n");} program_commands ;

default : T_DEFAULT T_colon program_commands ;
```

Στις εντολές ανάθεσης όπου έχουμε την μορφή , ο κώδικας μας υποστηρίζει ως έκφραση μεταβλητές, νούμερα και σύνθετες παραστάσεις με οποιαδήποτε πράξη ανάμεσα σε μεταβλητές και αριθμούς. Δεν μπορέσαμε να υλοποιήσουμε ολοκληρωτικά την κλήση συνάρτησης και στις σύνθετες παραστάσεις τη χρήση παρενθέσεων.

2.1.β Συντακτικός και Λεκτικός αναλυτής

Λεκτικός Αναλυτής:

```
1
2   %option yylineno
3
4   %{
5   #include "bison.tab.h"
6   #include <stdio.h>
7   #include <stdlib.h>
8   #include <string.h>
9
10
11  int error_counter=0;
12  void yyerror(const char* err);
13  void printtokens(int tokenum);
14  %}
15
16  %option noyywrap
17  %option yylineno
18
19  %x C_COMMENT
20
21  letters [a-zA-z]
22  digit         [0-9]
23  intident      [1-9]
24
25  alphanums     ([a-zA-Z]+)[0-9]+
26  boolean      ("true"|"false")
27  ws           [ \t\n\r]
28  string   \"([^\\\"]|\\.)*\"
29  floatn    [+-]?([0-9]*[.])?[0-9]+
30
31
32  %%
33
34  "PROGRAM"       {printtokens(T_PROGRAM); return T_PROGRAM;}
35  "FUNCTION"     {printtokens(T_FUNCTION); return T_FUNCTION;}
36  "PRINT"       {printtokens(T_PRINT); return T_PRINT;}
37  "VARS"         {printtokens(T_VARS); return T_VARS;}
38  "CHAR"         {printtokens(T_char); return T_char;}
39  "ENDFUNCTION" {printtokens(T_end); return T_end;}
40  "RETURN"       {printtokens(T_return); return T_return;}
41  "STARTMAIN"     {printtokens(T_STARTMAIN); return T_STARTMAIN;}
42  "ENDMAIN"       {printtokens(T_ENDMAIN); return T_ENDMAIN;}
43  "IF"           {printtokens(T_IF); return T_IF;}
44  "TYPEDEF"           {printtokens(T_TYPEDEF); return T_TYPEDEF;}
45  "STRUCT"           {printtokens(T_STRUCT); return T_STRUCT;}
46  "ENDSTRUCT"             {printtokens(T_ENDSTRUCT); return T_ENDSTRUCT;}
47  "SWITCH"       {printtokens(T_SWITCH); return T_SWITCH;}
48  "CASE"         {printtokens(T_CASE); return T_CASE;}
49  "BREAK"        {printtokens(T_BREAK); return T_BREAK;}
50  "WHILE"        {printtokens(T_WHILE); return T_WHILE;}
51  "INT"          {printtokens(T_int); return T_int;}
52  "FLOAT"       {printtokens(T_float); return T_float;}
53  "ELSE"         {printtokens(T_ELSE); return T_ELSE;}
54  "ELSEIF"       {printtokens(T_ELSEIF); return T_ELSEIF;}
55  "FOR"          {printtokens(T_FOR); return T_FOR;}
```

```
"=="|"!="       {printtokens(T_equop); return T_equop;}
">"|"<"|">="|"<=" {printtokens(T_relop); return T_relop;}
"+"|"-"       {printtokens(T_adop); return T_adop;}
"*"|"/"|"%"|"^"  {printtokens(T_mulop); return T_mulop;}
"||"          {printtokens(T_OROP); return T_OROP;}
"&&"          {printtokens(T_ANDOP); return T_ANDOP;}
"."           {printtokens(T_telia); return T_telia ;}
";"           {printtokens(T_semicolon); return T_semicolon ;}
":"                 {printtokens(T_colon); return T_colon;}
","                 {printtokens(T_komma); return T_komma;}
"{"                   {printtokens(T_openagk); return T_openagk;}
"="           {printtokens(T_ASSIGN); return T_ASSIGN;}
"}"                   {printtokens(T_closeagk); return T_closeagk;}
"("           {printtokens(T_open); return T_open;}
")"           {printtokens(T_close); return T_close;}
"["                   {printtokens(T_openpar); return T_openpar;}
"]"                   {printtokens(T_closepar); return T_closepar;}
"STEP"        {printtokens(T_STEP); return T_STEP;}
"TO"           {printtokens(T_TO); return T_TO;}
"ENDFOR"       {printtokens(T_ENDFOR); return T_ENDFOR;}
"ENDWHILE"      {printtokens(T_ENDWHILE); return T_ENDWHILE;}
"ENDSWITCH"      {printtokens(T_ENDSWITCH); return T_ENDSWITCH;}
"ENDIF"      {printtokens(T_ENDIF); return T_ENDIF;}
"THEN"      {printtokens(T_THEN); return T_THEN;}
"DEFAULT"      {printtokens(T_DEFAULT); return T_DEFAULT;}
{boolean}       {printtokens(T_boolean); return T_boolean;}
{floatn}               {printtokens(T_floatident); return T_floatident;}
{intident}*{digit}*|0     {printtokens(T_intident ); return T_intident ;}
{alphanums}|{letters}+  {printtokens(T_charident ); return T_charident ;}
{alphanums}"["{digit}+"]"|{letters}+"["{digit}+"]" {printtokens(T_pin); return T_pin;}
{string}  {printtokens(T_string ); return T_string ;}
"/*"            { BEGIN(C_COMMENT); }
<C_COMMENT>"*/" { BEGIN(INITIAL); }
<C_COMMENT>.    { }
[%].*


{ws}                {     }
.           {yyerror("unrecognized character");}



<<EOF>>  {printtokens(T_eof); return T_eof;}

%%


```

```
void printtokens(int tokennum){
  printf("found token \'%s\' (%d) at line %d \n",yytext,tokennum,yylineno);
}

void yyerror(const char* err){
  error_counter++;
  printf("[error - LINE %d]%s\n",yylineno,err );
  if(error_counter==10){
    printf("found max errors");
  }

}
```

Συντακτικός αναλυτής:

```
1
2   %{
3   #include <stdio.h>
4   #include <math.h>
5   #include <stdlib.h>
6
7
8   extern void yyerror(const char* err);
9   extern FILE *yyin;
10  extern FILE *yyout;
11  extern int yylex();
12    extern int yyparse();
13      extern int yylineno;
14  %}
15
16  %error-verbose
17
18  %union{
19      int intval;
20      float floatval;
21      char charval;
22  }
23
24  %token T_eof   0 "end of file"
25  %token T_IF   "IF"
26  %token T_ELSE  "ELSE"
27  %token T_STARTMAIN "STARTMAIN"
28  %token T_FOR   "FOR"
29  %token T_FUNCTION  "FUNCTION"
30  %token T_PROGRAM  "PROGRAM"
31  %token T_semicolon  ";"
32  %token T_komma ","
33  %token T_openagk "{"
34  %token T_closeagk "}"
35  %token T_openpar "["
36  %token T_closepar "]"
37  %token T_boolean "true or false"
38  %token <intval> T_intident "intnumber"
39  %token T_int "INT"
40  %token T_VARS "VARS"
41  %token T_float "FLOAT"
42  %token <floatval> T_floatident "floatnumber"
43  %token T_ws "kena"
44  %token T_telia "."
45  %token T_ELSEIF "ELSEIF"
46  %token T_ANDOP "&&"
47  %token T_OROP "||"
48  %token T_NOT "!"
49  %token T_adop  "+ or -"
50  %token T_equop " == or != "
51  %token T_end "END_FUNCTION"
52  %token T_mulop "* or / or  % or ^"
53  %token T_char "CHAR"
54  %token T_ENDMAIN "ENDMAIN"
55  %token T_return "RETURN"
```

```
%token <charval> T_charident "charnumber"
%token T_pin "TABLE"
%token T_open "("
%token T_close ")"
%token T_relop "< or > or <= or >="
%token T_ASSIGN "="
%token T_TO "TO"
%token T_STEP "STEP"
%token T_BREAK "BREAK"
%token T_ENDFOR "ENDFOR"
%token T_SWITCH "SWITCH"
%token T_WHILE "WHILE"
%token T_colon ":"
%token T_CASE "CASE" /**/
%token T_PRINT "PRINT"
%token T_ENDWHILE "ENDWHILE"
%token T_ENDIF "ENDIF"
%token T_THEN "THEN"
%token T_ENDSWITCH "ENDSWITCH"
%token T_DEFAULT "DEFAULT"
%token T_STRUCT "STRUCT"
%token T_ENDSTRUCT "ENDSTRUCT"
%token T_TYPEDEF "TYPEDEF"
%token T_string "string"

%%

PROGRAM: T_PROGRAM name {printf("\n");} body_program ;

name: T_charident ;

Struct : T_STRUCT name {printf("\n");} variable_declaration T_ENDSTRUCT
  |T_TYPEDEF T_STRUCT name {printf("\n");} variable_declaration {printf("\n");} name T_ENDSTRUCT
  ;

body_program : Struct Functions  Main_function
   | Struct Main_function
   | Main_function
   ;

Functions : T_FUNCTION name T_open P1 T_close body_Function T_return T_charident {printf("\n");} T_end ;

P1 : %empty
   | V1 P1
   ;

V1 : type_variable d
   | type_variable d T_komma V1
   ;

d  : name
   | table
   ;

table :T_pin ;
```

```
V1 : type_variable d
   | type_variable d T_komma V1
   ;

d  : name
   | table
   ;

table :T_pin ;

type_variable : T_int
              | T_float
              | T_char
              ;

body_Function : variable_declaration program_commands
              | variable_declaration
              | %empty
              ;

variable_declaration : T_VARS format  T_semicolon ; // format = tropos dilosis
```

```
format : type_variable P2 ;

P2 : V2
   | V2 P2
   ;
V2 : d
   | d T_komma | T_komma d G
   ;

G  : %empty
   ;

program_commands : assign_commands extra_commands
                 | assign_commands
                 | assign_commands program_commands
                 | extra_commands
                 | %empty
                 ;


assign_commands: variable T_ASSIGN expression T_semicolon ;

expression:  oros
          | function
          ;

function: name T_open orisma T_close  ;

orisma: variable
      | variable T_komma orisma
      ;

oros: A
    | A telestes A oros
    | A telestes parenthesi oros
    ;

parenthesi: T_open A T_close
          | T_open A T_close parenthesi ;

variable : d ;

A: variable
  | variable telestes A
  | number
  | number telestes A
  ;

 telestes: T_adop
         | T_mulop
         ;

number : T_intident
       | T_floatident
```

```
178         ;
179
180 Main_function: T_STARTMAIN variable_declaration program_commands T_ENDMAIN
181             |  T_STARTMAIN program_commands T_ENDMAIN
182             ;
183
184 extra_commands : H ;
185
186
187 H : Q
188   |Q H
189   ;
190
191 Q : loop_commands control_commands print_command
192               | loop_commands print_command
193               | control_commands print_command
194               | loop_commands  control_commands
195               | print_command
196               | loop_commands
197               | control_commands
198               ;
199
200
201 loop_commands : For_command
202                | While_command
203                ;
204
205 For_command : T_FOR counter T_colon T_ASSIGN number T_TO number T_STEP number {printf("\n");} program_commands L T_ENDFOR ;
206
207 //type :T_intident
208      | T_charident ;//
209
210
211
212 counter : variable ;
213
214 While_command : T_WHILE T_open condition T_close {printf("\n");} program_commands L T_ENDWHILE ;
215
216
217 comparison_operator: T_relop
218  | T_equop
219  ;
220
221 logic_operators: T_ANDOP
222  | T_OROP
223  ;
224
225 L: %empty
226  | break_command L
227  ;
228
229 break_command : T_BREAK T_semicolon ;
```

```
print_command : T_PRINT T_open T_string X T_close T_semicolon ;

X : %empty
  | F X  ;

F :   T_komma d  ;


control_commands: If_command
                | Switch_command
                | If_command Switch_command
                ;

If_command : T_IF T_open condition T_close T_THEN {printf("\n");} program_commands elseif else L T_ENDIF ;

condition : P
          | T
          ;

 P: A comparison_operator A ;

 T: %empty
  | J T
  ;

 J: P logic_operators P
  | logic_operators P
  ;


elseif : %empty
      | T_ELSEIF  program_commands elseif
      ;

else : %empty
      | T_ELSE program_commands
      ;

Switch_command : T_SWITCH  T_open expression T_close {printf("\n");} case  T_ENDSWITCH ;

case : M L
     | M L case
     | M L default L
     ;

M : T_CASE T_open expression T_close T_colon {printf("\n");} program_commands ;

default : T_DEFAULT T_colon program_commands ;

%%
```

```
int main(int argc, char *argv[]) {
      int token;
      if(argc>1){
        yyin=fopen(argv[1],"r");
        if(yyin==NULL){
          perror ("error open");
          return -1;
        }
      }

            yyparse();

      fclose(yyin);
      return 0;
}
```

2.2 Δήλωση τύπου δεδομένων χρήστη / Δήλωση δομής

Η δήλωση τύπου δεδομένων χρήστη / δήλωση δομής πραγματοποιείται μετά τον προαιρετικό ορισμό των συναρτήσεων.

Για την κατάλληλη υλοποίηση του συγκεκριμένου ερωτήματος έγινε μετατροπή/προσθήκη στον αναλυτή:

```
91
92  Struct : T_STRUCT name {printf("\n");} variable_declaration T_ENDSTRUCT
93         |T_TYPEDEF T_STRUCT name {printf("\n");} variable_declaration {printf("\n");} name T_ENDSTRUCT
94         ;
95
```

2.3 Εμφάνιση σφάλματος

Εμφάνιση/Διόρθωση σφάλματος Για τον έλεγχο της σωστής δήλωσης των μεταβλητών και των συναρτήσεων που χρησιμοποιούνται οπουδήποτε στο πρόγραμμα , εμφανίζεται σχετικό μήνυμα σφάλματος το οποίο διακόπτει τη διαδικασία της ανάλυσης μέσω της yyerror και σου εμφανίζει τον τρόπο αντικατάστασης του λάθους μέσω του %error-verbose.

```
15
16  %error-verbose
17
```

2.4Σχόλια

Μετά από κατάλληλες αλλαγές ο κώδικας μας υποστηρίζει τα σχόλια πολλαπλής γραμμής της μορφής :

/*… σχόλια

…

σχόλια */

ή

/* σχόλια */

```
21
22  %x C_COMMENT
23
```

```
90  "/*"              { BEGIN(C_COMMENT); }
91  <C_COMMENT>"*/"   { BEGIN(INITIAL); }
92  <C_COMMENT>.      { }
93  [%].*
```

3. INPUT

3.1) Δημιουργήσαμε 2 input για να δοκιμάσουμε των κώδικα μας :

Input2 :

```
1  PROGRAM EXAMPLE2
2  TYPEDEF STRUCT example2
3
4  VARS
5  CHAR var1,var2[4],var3;
6  example2 ENDSTRUCT
7
8  FUNCTION example2(INT num, CHAR var1)
9  VARS
10 CHAR var5;
11 RETURN var5
12 END_FUNCTION
13
14 STARTMAIN
15
16
17 var6=var1;
18 var2=1;
19 var4=var1/ 1 + 2*var1- 4+ 2 * var2[5] ;
20
21 FOR counter:=10 TO 1000 STEP 10
22 WHILE ( var1==num && var2<2 || var3<4 )
23 PRINT("apotelesma=num");
24 ENDWHILE
25 ENDFOR
26 ENDMAIN
```

Input1:

```
PROGRAM EXAMPLE1
STRUCT example
VARS
CHAR var[10];
ENDSTRUCT

FUNCTION example1( CHAR x, INT z)
VARS/* DILOSI shagv
hsahbxcav */
CHAR var1,var2,var3,var[10];
var2=12;
var3=10;

RETURN var3
END_FUNCTION


STARTMAIN
VARS
CHAR var7;
var2=12;

IF(var1>5)
THEN
PRINT("Result=%d",var1);
ELSEIF
PRINT("Result=%d",var3);
ELSE
PRINT("Result=%d",var2);
ENDIF

SWITCH(num)
CASE(var3):
PRINT("apotelesma=10");
BREAK;
CASE(var2):
PRINT("apotelesma=12");
BREAK;
DEFAULT:
PRINT("apotelesma=%d", num);
BREAK;

ENDSWITCH
ENDMAIN
```

3.2 ΑΠΟΤΕΛΕΣΜΑΤΑ

Το αποτέλεσμα από το input2 :

```
BESIANA@DESKTOP-HR041DD /cygdrive/c/Cygwin64/plta
$ ./a.exe input2
found token 'PROGRAM' (263) at line 1
found token 'EXAMPLE2' (289) at line 1

found token 'TYPEDEF' (311) at line 2
found token 'STRUCT' (309) at line 2
found token 'example2' (289) at line 2

found token 'VARS' (273) at line 4
found token 'CHAR' (286) at line 5
found token 'var1' (289) at line 5
found token ',' (265) at line 5
found token 'var2[4]' (290) at line 5
found token ',' (265) at line 5
found token 'var3' (289) at line 5
found token ';' (264) at line 5

found token 'example2' (289) at line 6
found token 'ENDSTRUCT' (310) at line 6
found token 'FUNCTION' (262) at line 8
found token 'example2' (289) at line 8
found token '(' (291) at line 8
found token 'INT' (272) at line 8
found token 'num' (289) at line 8
found token ',' (265) at line 8
found token 'CHAR' (286) at line 8
found token 'var1' (289) at line 8
found token ')' (292) at line 8
found token 'VARS' (273) at line 9
found token 'CHAR' (286) at line 10
found token 'var5' (289) at line 10
found token ';' (264) at line 10
found token 'RETURN' (288) at line 11
found token 'var5' (289) at line 11

found token 'END_FUNCTION' (284) at line 12
found token 'STARTMAIN' (260) at line 14
found token 'var6' (289) at line 17
found token '=' (294) at line 17
found token 'var1' (289) at line 17
found token ';' (264) at line 17
found token 'var2' (289) at line 18
found token '=' (294) at line 18
found token '1' (275) at line 18
found token ';' (264) at line 18
found token 'var4' (289) at line 19
found token '=' (294) at line 19
found token 'var1' (289) at line 19
found token '/' (285) at line 19
found token '1' (275) at line 19
found token '+' (282) at line 19
found token '2' (275) at line 19
found token '*' (285) at line 19
found token 'var1' (289) at line 19
found token '-' (282) at line 19
found token '4' (275) at line 19
found token '+' (282) at line 19
found token '2' (275) at line 19
```

```
found token '*' (285) at line 19
found token 'var2[5]' (290) at line 19
found token ';' (264) at line 19
found token 'FOR' (261) at line 21
found token 'counter' (289) at line 21
found token ':' (301) at line 21
found token '=' (294) at line 21
found token '10' (275) at line 21
found token 'TO' (295) at line 21
found token '1000' (275) at line 21
found token 'STEP' (296) at line 21
found token '10' (275) at line 21

found token 'WHILE' (300) at line 22
found token '(' (291) at line 22
found token 'var1' (289) at line 22
found token '==' (283) at line 22
found token 'num' (289) at line 22
found token '&&' (279) at line 22
found token 'var2' (289) at line 22
found token '<' (293) at line 22
found token '2' (275) at line 22
found token '||' (280) at line 22
found token 'var3' (289) at line 22
found token '<' (293) at line 22
found token '4' (275) at line 22
found token ')' (292) at line 22

found token 'PRINT' (303) at line 23
found token '(' (291) at line 23
found token '"apotelesma=num"' (312) at line 23
found token ')' (292) at line 23
found token ';' (264) at line 23
found token 'ENDWHILE' (304) at line 24
found token 'ENDFOR' (298) at line 25
found token 'ENDMAIN' (287) at line 26
found token '' (0) at line 27

BESIANA@DESKTOP-HR041DD /cygdrive/c/Cygwin64/plta
$
```

Τα αποτελέσματα από το input1:

```
BESIANA@DESKTOP-HR041DD /cygdrive/c/Cygwin64/plta
$ ./a.exe input1
found token 'PROGRAM' (263) at line 1
found token 'EXAMPLE1' (289) at line 1

found token 'STRUCT' (309) at line 2
found token 'example' (289) at line 2

found token 'VARS' (273) at line 3
found token 'CHAR' (286) at line 4
found token 'var[10]' (290) at line 4
found token ';' (264) at line 4
found token 'ENDSTRUCT' (310) at line 5
found token 'FUNCTION' (262) at line 7
found token 'example1' (289) at line 7
found token '(' (291) at line 7
found token 'CHAR' (286) at line 7
found token 'x' (289) at line 7
found token ',' (265) at line 7
found token 'INT' (272) at line 7
found token 'z' (289) at line 7
found token ')' (292) at line 7
found token 'VARS' (273) at line 8
```

```
found token 'CHAR' (286) at line 9
found token 'var1' (289) at line 9
found token ',' (265) at line 9
found token 'var2' (289) at line 9
found token ',' (265) at line 9
found token 'var3' (289) at line 9
found token ',' (265) at line 9
found token 'var[10]' (290) at line 9
found token ';' (264) at line 9
found token 'var2' (289) at line 10
found token '=' (294) at line 10
found token '12' (275) at line 10
found token ';' (264) at line 10
found token 'var3' (289) at line 11
found token '=' (294) at line 11
found token '10' (275) at line 11
found token ';' (264) at line 11
found token 'RETURN' (288) at line 13
found token 'var3' (289) at line 13

found token 'END_FUNCTION' (284) at line 14
found token 'STARTMAIN' (260) at line 17
found token 'VARS' (273) at line 18
found token 'CHAR' (286) at line 19
found token 'var7' (289) at line 19
found token ';' (264) at line 19
found token 'var2' (289) at line 20
found token '=' (294) at line 20
found token '12' (275) at line 20
found token ';' (264) at line 20
found token 'IF' (258) at line 22
found token '(' (291) at line 22
found token 'var1' (289) at line 22
found token '>' (293) at line 22
found token '5' (275) at line 22
found token ')' (292) at line 22
found token 'THEN' (306) at line 23

found token 'PRINT' (303) at line 24
found token '(' (291) at line 24
found token '"Result=%d"' (312) at line 24
found token ',' (265) at line 24
found token 'var1' (289) at line 24
found token ')' (292) at line 24
found token ';' (264) at line 24
found token 'ELSEIF' (278) at line 25
found token 'PRINT' (303) at line 26
found token '(' (291) at line 26
found token '"Result=%d"' (312) at line 26
found token ',' (265) at line 26
found token 'var3' (289) at line 26
found token ')' (292) at line 26
found token ';' (264) at line 26
found token 'ELSE' (259) at line 27
found token 'PRINT' (303) at line 28
found token '(' (291) at line 28
found token '"Result=%d"' (312) at line 28
```

```
found token ',' (265) at line 26
found token 'var3' (289) at line 26
found token ')' (292) at line 26
found token ';' (264) at line 26
found token 'ELSE' (259) at line 27
found token 'PRINT' (303) at line 28
found token '(' (291) at line 28
found token '"Result=%d"' (312) at line 28
found token ',' (265) at line 28
found token 'var2' (289) at line 28
found token ')' (292) at line 28
found token ';' (264) at line 28
found token 'ENDIF' (305) at line 29
found token 'SWITCH' (299) at line 31
found token '(' (291) at line 31
found token 'num' (289) at line 31
found token ')' (292) at line 31

found token 'CASE' (302) at line 32
found token '(' (291) at line 32
found token 'var3' (289) at line 32
found token ')' (292) at line 32
found token ':' (301) at line 32

found token 'PRINT' (303) at line 33
found token '(' (291) at line 33
found token '"apotelesma=10"' (312) at line 33
found token ')' (292) at line 33
found token ';' (264) at line 33
found token 'BREAK' (297) at line 34
found token ';' (264) at line 34
found token 'CASE' (302) at line 35
found token '(' (291) at line 35
found token 'var2' (289) at line 35
found token ')' (292) at line 35
found token ':' (301) at line 35

found token 'PRINT' (303) at line 36
found token '(' (291) at line 36
found token '"apotelesma=12"' (312) at line 36
found token ')' (292) at line 36
found token ';' (264) at line 36
found token 'BREAK' (297) at line 37
found token ';' (264) at line 37
found token 'DEFAULT' (308) at line 38
found token ':' (301) at line 38
found token 'PRINT' (303) at line 39
found token '(' (291) at line 39
found token '"apotelesma=%d"' (312) at line 39
found token ',' (265) at line 39
found token 'num' (289) at line 39
found token ')' (292) at line 39
found token ';' (264) at line 39
found token 'BREAK' (297) at line 40
found token ';' (264) at line 40
found token 'ENDSWITCH' (307) at line 42
found token 'ENDMAIN' (287) at line 43
found token '' (0) at line 43

BESIANA@DESKTOP-HR041DD /cygdrive/c/Cygwin64/plta
$
```