Αρχές Γλωσσών Προγραμματισμού & Μεταφραστών

Εργαστηριακή Άσκηση 2020-2021

Μέλη Ομάδας:

Φωτεινός Εμμανουήλ ΑΜ: 1067428
3ο Έτος manolistg5000@hotmail.com

Χρυσανθοπούλου Αριστέα ΑΜ: 1067483
3ο Έτος up1067483@upnet.gr

Θεοδώρου Μιχαήλ ΑΜ: 1067391
(3ο Έτος up1067391@upnet.gr)

Μαρκοδήμος Παναγιώτης ΑΜ: 1067523
3ο Έτος p3t3marks@gmail.com

Περιγραφή της γραμματικής της γλώσσας σε BNF:

```
program::=      T_PROGRAM T_ID T_NL <structs> T_NL <functions> T_NL <main>
                | T_PROGRAM T_ID T_NL <functions> T_NL <main>
                | T_PROGRAM T_ID T_NL <structs> T_NL <main>
                | T_PROGRAM T_ID T_NL <main>

<functions> ::= <function>
                | <functions> T_NL <function>

<function> ::=  T_FUNCTION T_ID T_LPAR <arguments> T_RPAR T_NL <body>

<body> ::=      <declare> <commands> <return> T_ENDFUNCTION

<declare> ::=   <declare>  T_VARS <type> <variable> T_SCOLON
                | ε

<return> ::=    T_RETURN T_NUMBER
                | T_RETURN T_C
                | T_RETURN T_ID

<variable> ::=  T_ID
                |T_ID T_LBRAC T_NUMBER T_RBRAC
                | <variable> T_COMMA <variable>

<type> ::=      T_CHAR
                | T_INTEGER
```

<main> ::= T_STARTMAIN <declare> <commands> T_ENDMAIN

<commands> ::= <commands> <command>
| ε

<command>::= command_assign
| command_loop
| command_check
| print
| break
| functioncall

<command_assign> ::= T_ID T_ASSIGN <expression> T_SCOLON
| T_ID T_ASSIGN T_C T_SCOLON

<expression> ::= <term>
| <expression> <operator> <term>

<term> ::= <factor>
| <functioncall>

<factor> ::= T_ID
| T_ID T_LBRAC T_NUMBER T_RBRAC
| T_NUMBER
| T_LPAR <expression> T_RPAR

<operator> ::= T_ADD
|T_SUB
|T_MUL
|T_DIV
|T_POWER

<functioncall> ::= T_ID T_LPAR <functioncall_arguments> T_RPAR

<functioncall_arguments>::= <factor>
| <functioncall_arguments> T_COMMA <functioncall_arguments>
| ε

<arguments> ::= <argument>
| <arguments> T_COMMA <argument>
| ε

<argument> ::= T_VARS <type> T_ID

<command_loop> ::= T_WHILE T_LPAR <condition> T_RPAR <commands> T_ENDWHILE
| T_FOR <count> T_TO T_NUMBER T_STEP T_NUMBER <commands>T_ENDFOR

<count> ::= T_ID T_COLON T_ASSIGN T_NUMBER

<condition> ::= <expression> <condition_operator> <expression>

<compare_operator> ::= T_LT
| T_GT
| T_EQ
| T_NQ
| T_AND
| T_OR

<command_check> ::= T_IF T_LPAR <condition> T_RPAR T_THEN <commands> <if_tail> T_ENDIF
| T_SWITCH T_LPAR <expression> T_RPAR <cases> T_ENDSWITCH

<cases> ::= <cases> <case>
| ε

case::= T_CASE T_LPAR <expression> T_RPAR T_COLON <commands>
| T_DEFAULT T_COLON <commands>

<if_tail> ::= <if_tail> T_ELSEIF T_LPAR <condition> T_RPAR <commands>
| <if_tail> T_ELSE <commands>
| ε

<print> ::= T_PRINT T_LPAR T_STRING T_COMMA T_LBRAC T_ID T_RBRAC T_RPAR T_SCOLON
|T_PRINT T_LPAR T_STRING T_PAR T_SCOLON

<break> ::= T_BREAK T_SCOLON

<structs>::= <struct>
| <structs> T_NL <struct>

<struct>::= T_STRUCT T_ID T_NL <declare> T_ENDSTRUCT
| T_TYPEDEF T_STRUCT T_ID T_NL <declare> T_ID T_ENDSTRUCT

Τελικό αρχείο περιγραφής λεκτικού αναλυτή:

```
%{
#include "bis.tab.h"
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <math.h>
extern YYSTYPE yylval;
//DEBUG = 1 για debug mode, εκτελούνται print, = 0 για απενεργοποίηση
#define DEBUG 1
//Όταν είναι 1, ο λεκτικός αναλυτής λαμβάνει υπόψη το new line. Γίνεται 1 μόνο όταν διαβαστούν συγκεκριμένα tokens.
int nl_flag = 0;
void t_print(int number);
char ss[256];
%}
 // Επίλογες του flex για την διευκόλυνση του κώδικα(noyywrap-> Διαβάζει ένα αρχείο κάθε φορά, yylineno-> Διατηρεί τον αριθμό της τρέχων γραμμής)
%option noyywrap
%option yylineno
 // Ορισμός Υποαναλυτών
%x MULTI_LINE_COMMENT
%x SINGLE_LINE_COMMENT
%x STRING
 // Ισοδύναμες εκφράσεις (Regular Expressions)
digit   [0-9]
num    {digit}+
id          [a-zA-Z]+[0-9a-zA-Z_]*
char       '[0-9a-zA-Z()!./,\?<>;:!'@#$%^&*+-=]'
```

```
%%
 //Κανόνες λεκτικής ανάλυσης
"PROGRAM"       {nl_flag = 1; t_print(T_PROGRAM); return T_PROGRAM;}
"FUNCTION"      {nl_flag = 1; t_print(T_FUNCTION); return T_FUNCTION;}
"END_FUNCTION"  {nl_flag = 1; t_print(T_ENDFUNCTION); return T_ENDFUNCTION;}
"VARS"          {t_print(T_VARS); return T_VARS;}
"CHAR"          {t_print(T_CHAR); return T_CHAR;}
"INTEGER"       {t_print(T_INTEGER); return T_INTEGER;}
"RETURN"        {t_print(T_RETURN); return T_RETURN;}
"STARTMAIN"     {t_print(T_STARTMAIN); return T_STARTMAIN;}
"ENDMAIN"       {t_print(T_ENDMAIN); return T_ENDMAIN;}
"WHILE"         {t_print(T_WHILE); return T_WHILE;}
"ENDWHILE"      {t_print(T_ENDWHILE); return T_ENDWHILE;}
"FOR"           {t_print(T_FOR); return T_FOR;}
"TO"            {t_print(T_TO); return T_TO;}
"STEP"          {t_print(T_STEP); return T_STEP;}
"ENDFOR"        {t_print(T_ENDFOR); return T_ENDFOR ;}
"IF"            {t_print(T_ENDIF); return T_IF;}
"THEN"          {t_print(T_THEN); return T_THEN;}
"ELSE"          {t_print(T_ELSE); return T_ELSE;}
"ELSEIF"        {t_print(T_ELSEIF); return T_ELSEIF;}
"ENDIF"         {t_print(T_ENDIF); return T_ENDIF;}
"SWITCH"        {t_print(T_SWITCH); return T_SWITCH;}
"CASE"          {t_print(T_CASE); return T_CASE;}
"DEFAULT"       {t_print(T_DEFAULT); return T_DEFAULT;}
"ENDSWITCH"     {t_print(T_ENDSWITCH); return T_ENDSWITCH;}
"PRINT"         {t_print(T_PRINT); return T_PRINT;}
"BREAK"         {t_print(T_BREAK); return T_BREAK;}
"STRUCT"        {nl_flag = 1; t_print(T_STRUCT); return T_STRUCT;}
"TYPEDEF"       {t_print(T_TYPEDEF); return T_TYPEDEF;}
"ENDSTRUCT"     {nl_flag = 1; t_print(T_ENDSTRUCT); return T_ENDSTRUCT;}

        /*MULTI_LINE_COMMENT*/
"/*"                    {printf("Multi-Lined Comment found\n"); BEGIN(MULTI_LINE_COMMENT);}
/*Είσοδος στον υποαναλυτή*/
<MULTI_LINE_COMMENT>"*/"    {BEGIN(INITIAL); printf("Multi-Lined Comment ended\n");}
/*Επιστροφή από τον υποαναλυτή*/
<MULTI_LINE_COMMENT>(\n|\r\n)   {}
<MULTI_LINE_COMMENT>(\.)    {}

        /*SINGLE_LINE_COMMENT*/
"%"                     {printf("Comment found\n"); BEGIN(SINGLE_LINE_COMMENT);}
/*Είσοδος στον υποαναλυτή*/
<SINGLE_LINE_COMMENT>(\n|\r\n)  {BEGIN(INITIAL); printf("Comment ended.\n");}          /*Επιστροφή
από τον υποαναλυτή*/
<SINGLE_LINE_COMMENT>.      {}                                  /*νεα γραμμη στο comment */

        /*STRING*/
\"                  {strcpy(ss, ""); printf("String found\n"); BEGIN(STRING);}          /*Είσοδος στον
υποαναλυτή*/
<STRING>\"      {BEGIN(INITIAL); yylval.strval=strdup(ss); printf("%s\n", ss); return T_STRING;}
/*Επιστροφή από τον υποαναλυτή*/
<STRING>(\n|\r\n)       {}                          /*νεα γραμμη στο string */
<STRING>.               {strcat(ss, yytext);}

        /*Τελεστές*/
"+"   {t_print(T_ADD); return T_ADD;}
```

```
"-"         {t_print(T_SUB); return T_SUB;}
"^"         {t_print(T_POWER); return T_POWER;}
"*"         {t_print(T_MUL); return T_MUL;}
"/"         {t_print(T_DIV); return T_DIV;}
">"         {t_print(T_GT); return T_GT;}
"<"         {t_print(T_LT); return T_LT;}
"=="        {t_print(T_EQ); return T_EQ;}
"!="        {t_print(T_NQ); return T_NQ;}
"AND"   {t_print(T_AND); return T_AND;}
"OR"        {t_print(T_OR); return T_OR;}


            /*Σύμβολα*/
"="         {t_print(T_ASSIGN); return T_ASSIGN;}
"("         {t_print(T_LPAR); return T_LPAR;}
")"         {t_print(T_RPAR); return T_RPAR;}
";"         {t_print(T_SCOLON); return T_SCOLON;}
","         {t_print(T_COMMA); return T_COMMA;}
":"         {t_print(T_COLON); return T_COLON;}
"["         {t_print(T_LBRAC); return T_LBRAC;}
"]"         {t_print(T_RBRAC); return T_RBRAC;}
[\r\n]+    {if(nl_flag==1) {t_print(T_NL); nl_flag=0; return T_NL;}}
[ \t]   {     }

{num}     {t_print(T_NUMBER); yylval.intval=atoi(yytext); return T_NUMBER;}    /* yylval-> Αποθηκεύει τις τιμές στο union του bis.y*/
{id}        {t_print(T_ID); yylval.strval=strdup(yytext); return T_ID;}          /* (intval, strval)-> Τύποι αποθήκευσης τιμών του union Integer και String(char *) αντίστοιχα*/
{char}  {t_print(T_C); yylval.strval-strdup(yytext); return T_C;}               /* atoi-> μετατρέπει ένα string σε integer, strdup->αντιγράφει ενα string και επιστρέφει τον δείκτη του*/

.           {t_print(Unknown); return Unknown;}

<<EOF>> {return T_EOF;}


%%
//Print που χρησιμοποιείται όταν κάνουμε debug
void t_print(int number){
            if(DEBUG){
            printf("%s | %d | %d\n", yytext, number, yylineno);
            }
}
```

Τελικό αρχείο περιγραφής συνακτικού αναλυτή:

```
%{
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include "bis.tab.h"
```

```
#include "hashtbl.h"
#include "hashtbl.c"
#include "settings.h"

void yyerror(char const *S); //Συνάρτηση για εμφάνιση error
extern int yylineno; //Καταμέτρηση γραμμών
extern char* yytext; //Χρήση yytext του lex
extern FILE *yyin; //Χρήση yyin του lex
extern FILE *yyout; //Χρήση yyout του lex
int yylex(void);
void typedef_id_match(char *x, char *y);
int scope=0;
HASHTBL *hashtbl;
%}
//Χρησιμοποιείται για διευκρίνιση των σφαλμάτων
%define parse.error verbose
//Ορισμός σωρών για τους τύπους δεδομένων των σημασιολογικών τιμών
%union{
      int intval;
      char *strval;
}
//Ορισμός των tokens
%token <strval> T_PROGRAM       "program"
%token <strval> T_FUNCTION      "function"
%token <strval> T_ENDFUNCTION   "endfunction"
%token <strval> T_STRUCT        "struct"
%token <strval> T_ENDSTRUCT     "endstruct"
%token <strval> T_TYPEDEF       "typedef"
%token <strval> T_VARS          "vars"
%token <strval> T_CHAR          "char"
%token <strval> T_INTEGER      "int"
%token <strval> T_RETURN        "return"
%token <strval> T_STARTMAIN     "startmain"
%token <strval> T_ENDMAIN       "endmain"

/*Εντολές*/
%token <strval> T_WHILE         "while"
%token <strval> T_ENDWHILE      "endwhile"
%token <strval> T_FOR           "for"
%token <strval> T_TO            "to"
%token <strval> T_STEP          "step"
%token <strval> T_ENDFOR        "endfor"
%token <strval> T_IF            "if"
```

```
%token <strval> T_THEN          "then"
%token <strval> T_ELSE          "else"
%token <strval> T_ELSEIF        "elseif"
%token <strval> T_ENDIF         "endif"
%token <strval> T_SWITCH        "switch"
%token <strval> T_CASE          "case"
%token <strval> T_DEFAULT       "default"
%token <strval> T_ENDSWITCH     "endswitch"
%token <strval> T_PRINT         "print"
%token <strval> T_BREAK         "break"
/*Symbols*/
%token <strval> T_ADD           "add"
%token <strval> T_SUB           "subtract"
%token <strval> T_POWER         "power"
%token <strval> T_MUL           "multiply"
%token <strval> T_DIV           "divide"
%token <strval> T_LT            "less than"
%token <strval> T_GT            "greater than"
%token <strval> T_EQ            "equal"
%token <strval> T_NQ            "not equal"
%token <strval> T_AND           "and"
%token <strval> T_OR            "or"
%token <strval> T_ASSIGN        "assign"
%token <strval> T_LPAR          "left parenthesis"
%token <strval> T_RPAR          "right parenthesis"
%token <strval> T_SCOLON        "semicolon"
%token <strval> T_COMMA         "comma"
%token <strval> T_COLON         "colon"
%token <strval> T_LBRAC         "left bracket"
%token <strval> T_RBRAC         "right bracket"
%token <strval> T_NL            "new line"
%token <strval> Unknown         "unknown"
 // Tokens δεδομένων
%token <strval> T_ID            "id"
%token <strval> T_STRING        "string"
%token <intval> T_NUMBER        "number"
%token <strval> T_C             "character"
/*End of File*/
%token <strval> T_EOF           0

//Προτεραιότητες συντελεστών
%left T_COMMA
%right T_ASSIGN
```

```
%left T_OR
%left T_AND
%left T_EQ T_NQ T_LT T_GT
%left T_ADD T_SUB
%left T_MUL T_DIV
%left T_POWER
%left T_LPAR T_RPAR T_LBRAC T_RBRAC
//Επίλυση dangling else
%nonassoc LOWER_THAN_ELSE
%nonassoc T_ELSE
%%


program:    T_PROGRAM T_ID T_NL structs T_NL functions T_NL main
{hashtbl_insert(hashtbl, $2, NULL, scope);}
            | T_PROGRAM T_ID T_NL functions T_NL main
{hashtbl_insert(hashtbl, $2, NULL, scope);}
            | T_PROGRAM T_ID T_NL structs T_NL main
{hashtbl_insert(hashtbl, $2, NULL, scope);}
            | T_PROGRAM T_ID T_NL main {hashtbl_insert(hashtbl, $2,
NULL, scope);}
            ;
functions:  function
            | functions T_NL function
            ;
function:   T_FUNCTION T_ID {hashtbl_insert(hashtbl, $2, NULL, scope);
scope++;} T_LPAR arguments T_RPAR T_NL body {scope--;}
            ;
body:       declare commands return T_ENDFUNCTION
            ;
declare:    declare  T_VARS type variable T_SCOLON
            | %empty
            ;
return:     T_RETURN T_NUMBER
            | T_RETURN T_C
            | T_RETURN T_ID   {hashtbl_search(hashtbl, $2, scope);}
            ;
variable:   T_ID {hashtbl_insert(hashtbl, $1, NULL, scope);}
            | T_ID T_LBRAC T_NUMBER T_RBRAC {hashtbl_insert(hashtbl,
$1, NULL, scope);}
            | variable T_COMMA variable
            ;
type:       T_CHAR
```

```
            | T_INTEGER
            ;
main:       T_STARTMAIN {scope++;} declare commands T_ENDMAIN
{scope--;}
            ;

commands:   commands  command
            | %empty
            ;
command:    command_assign
            | command_loop
            | command_check
            | print
            | break
            | functioncall
            ;
command_assign:   T_ID T_ASSIGN expression T_SCOLON
{hashtbl_search(hashtbl, $1, scope); }
                  | T_ID T_ASSIGN T_C T_SCOLON {hashtbl_search(hashtbl,
$1, scope);}
                  ;
expression: term
            | expression operator term
            ;
term:       factor
            | functioncall
            ;
factor:     T_ID {hashtbl_search(hashtbl, $1, scope);}
            | T_ID T_LBRAC T_NUMBER T_RBRAC {hashtbl_search(hashtbl,
$1, scope);}
            | T_NUMBER
            | T_LPAR expression T_RPAR
            ;
operator:   T_ADD
            |T_SUB
            |T_MUL
            |T_DIV
            |T_POWER
            ;
functioncall:     T_ID T_LPAR functioncall_arguments T_RPAR
{hashtbl_search(hashtbl, $1, scope);}
                  ;
functioncall_arguments: factor
```

```
                        | functioncall_arguments T_COMMA
functioncall_arguments
                        | %empty
                        ;
arguments:  argument
            | arguments T_COMMA argument
            | %empty
            ;
argument:   T_VARS type T_ID {hashtbl_insert(hashtbl, $3, NULL,
scope);}
            ;

command_loop:     T_WHILE T_LPAR condition T_RPAR {scope++;} commands
T_ENDWHILE {scope--;}
                  | T_FOR count T_TO T_NUMBER T_STEP T_NUMBER
{scope++;} commands T_ENDFOR {scope--;}
                  ;
count:      T_ID T_COLON T_ASSIGN T_NUMBER {hashtbl_search(hashtbl, $1,
scope);}
            ;
condition:  expression condition_operator expression
            ;
condition_operator:     T_LT
                        | T_GT
                        | T_EQ
                        | T_NQ
                        | T_AND
                        | T_OR
                        ;
command_check:    T_IF T_LPAR condition T_RPAR {scope++;} T_THEN
commands if_tail T_ENDIF {scope--;}
                  | T_SWITCH T_LPAR expression T_RPAR cases T_ENDSWITCH
                  ;
cases:            cases case
                  | %empty
                  ;

case:             T_CASE T_LPAR expression T_RPAR T_COLON {scope++;}
commands  {scope--;}
                  | T_DEFAULT T_COLON {scope++;} commands {scope--;}
                  ;
```

```
if_tail:          if_tail T_ELSEIF T_LPAR condition T_RPAR {scope++;}
commands {scope--;}
                  | if_tail T_ELSE {scope++;} commands {scope--;}
                  | %empty %prec LOWER_THAN_ELSE
                  ;
print:            T_PRINT T_LPAR T_STRING T_COMMA T_LBRAC T_ID T_RBRAC
T_RPAR T_SCOLON {hashtbl_insert(hashtbl, $3, NULL, scope);
hashtbl_search(hashtbl, $6, scope);}
                  | T_PRINT T_LPAR T_STRING T_RPAR  T_SCOLON
{hashtbl_insert(hashtbl, $3, NULL, scope);}
                  ;
break:            T_BREAK T_SCOLON
                  ;
structs:          struct
                  | structs T_NL struct
                  ;
struct:           T_STRUCT T_ID {scope++;} T_NL declare T_ENDSTRUCT
{hashtbl_insert(hashtbl, $2, NULL, scope); scope--;}
                  | T_TYPEDEF T_STRUCT T_ID {scope++;} T_NL declare
T_ID T_ENDSTRUCT {hashtbl_insert(hashtbl, $3, NULL, scope);
hashtbl_search(hashtbl, $7, scope); scope--;}
                  ;
%%

//Εμφάνιση error όταν υπάρχει συντακτικό λάθος
void yyerror(char const *s)
{
    fprintf(stderr,"\nThere is an error: %s\nLine: %d\nToken: %s\n", s,
yylineno, yytext);
      exit(1);
}

//Χρησιμοποιείται για να εξασφαλίσει ότι το όνομα του struct
χρησιμοποιείται σωστά στον ορισμό του με typedef
void typedef_id_match(char *x, char *y){
      int flag;
      flag = strcmp(x, y);
      if ( flag != 0)
      yyerror("Wrong use of TYPEDEF(ID's do not match)\n");

}
```

```c
int main ( int argc, char **argv  )
  {
      //Δημιουργία hash table για αποθήκευση τιμών
      if (!(hashtbl=hashtbl_create(10, NULL))){
        puts ("[Error]: Could not open file");
        return EXIT_FAILURE;
       }
  ++argv; --argc;
  if ( argc > 0 )
        yyin = fopen( argv[0], "r" );
  else
        yyin = stdin;
  yyout = fopen ( "output", "w" );
  yyparse ();
  fclose(yyin);
  //Αποδέσμευση μνήμης με καταστροφή του hash table
  hashtbl_destroy(hashtbl);
  return 0;
  }
```

Παραδείγματα:

Σε κάθε παράδειγμα δείχνουμε με την πρώτη εικόνα το input και με τις επόμενες δείχνουμε τι εμφανίζει το terminal. Στο terminal, εμφανίζεται ένα μονοδιάστατο πινακάκι για κάθε token. Η πρώτη στήλη εμφανίζει τι διαβάστηκε, η δεύτερη τον αριθμό που αντιστοιχεί στο συγκεκριμένο token και η τρίτη τον αριθμό γραμμής στην οποία διαβάστηκε

Program σωστό:

```
1    PROGRAM onoma_programmatos
2
```

```
PROGRAM | 258 | 1
onoma_programmatos | 308 | 1
```

Program χωρίς αλλαγή γραμμής:

```
1    PROGRAM onoma_programmatos TYPEDEF STRUCT tipos1
```

```
PROGRAM | 258 | 1
onoma_programmatos | 308 | 1
TYPEDEF | 263 | 1

There is an error: syntax error, unexpected typedef, expecting new line
Line: 1
Token: TYPEDEF
```

Struct σωστό:

```
3    TYPEDEF STRUCT tipos1
4        VARS INTEGER struct_var1;
5    tipos ENDSTRUCT
```

```
TYPEDEF | 263 | 3
STRUCT | 261 | 3
tipos1 | 308 | 3

 | 306 | 4
VARS | 264 | 4
INTEGER | 266 | 4
struct_var1 | 308 | 4
; | 301 | 4
HASHTBL_INSERT(): KEY = struct_var1, HASH = 0,          DATA = (null), SCOPE = 1
tipos1 | 308 | 5
ENDSTRUCT | 262 | 5
HASHTBL_INSERT(): KEY = tipos1, HASH = 8,       DATA = (null), SCOPE = 1
```

Struct λάθος:

```
TYPEDEF STRUCT tipos1
    VARS INTEGER struct_var1;
oxi_tipos1 ENDSTRUCT
```

```
TYPEDEF | 263 | 3
STRUCT | 261 | 3
tipos1 | 308 | 3

 | 306 | 4
VARS | 264 | 4
INTEGER | 266 | 4
struct_var1 | 308 | 4
; | 301 | 4
HASHTBL_INSERT(): KEY = struct_var1, HASH = 0,          DATA = (null), SCOPE = 1
oxi_tipos1 | 308 | 5
ENDSTRUCT | 262 | 5
HASHTBL_INSERT(): KEY = tipos1, HASH = 8,       DATA = (null), SCOPE = 1

There is an error|Token: oxi_tipos1 was not found
```

Function σωστό:

```
FUNCTION sinartisi(VARS INTEGER arg1, VARS CHAR arg2)
    VARS INTEGER var_int1,var_int2,var_int3[10];
    VARS CHAR var_char;
    var_int2 = 5;
    var_int1 = var_int2 * 3  + 2;
    RETURN arg1
END_FUNCTION
```

```
FUNCTION | 259 | 10
sinartisi | 308 | 10
HASHTBL_INSERT(): KEY = sinartisi, HASH = 2,    DATA = (null), SCOPE = 0
( | 299 | 10
VARS | 264 | 10
INTEGER | 266 | 10
arg1 | 308 | 10
HASHTBL_INSERT(): KEY = arg1, HASH = 3,         DATA = (null), SCOPE = 1
, | 302 | 10
VARS | 264 | 10
CHAR | 265 | 10
arg2 | 308 | 10
HASHTBL_INSERT(): KEY = arg2, HASH = 4,         DATA = (null), SCOPE = 1
) | 300 | 10

 | 306 | 11
VARS | 264 | 11
INTEGER | 266 | 11
var_int1 | 308 | 11
, | 302 | 11
HASHTBL_INSERT(): KEY = var_int1, HASH = 4,     DATA = (null), SCOPE = 1
var_int2 | 308 | 11
, | 302 | 11
HASHTBL_INSERT(): KEY = var_int2, HASH = 5,     DATA = (null), SCOPE = 1
var_int3 | 308 | 11
[ | 304 | 11
10 | 310 | 11
] | 305 | 11
HASHTBL_INSERT(): KEY = var_int3, HASH = 6,     DATA = (null), SCOPE = 1
; | 301 | 11
VARS | 264 | 12
CHAR | 265 | 12
var_char | 308 | 12
; | 301 | 12
HASHTBL_INSERT(): KEY = var_char, HASH = 8,     DATA = (null), SCOPE = 1
```

```
var_int2 | 308 | 13
= | 298 | 13
5 | 310 | 13
; | 301 | 13
var_int1 | 308 | 14
= | 298 | 14
var_int2 | 308 | 14
* | 290 | 14
3 | 310 | 14
+ | 287 | 14
2 | 310 | 14
; | 301 | 14
RETURN | 267 | 15
arg1 | 308 | 15
END_FUNCTION | 260 | 16
```

Function χωρίς declare, σωστό:

```
FUNCTION sinartisi()
    VARS INTEGER var_int1,var_int2,var_int3[10];
    VARS CHAR var_char;
    var_int1 = var_int2 * 3  + 2;
    RETURN 0
END_FUNCTION
```

```
FUNCTION | 259 | 10
sinartisi | 308 | 10
HASHTBL_INSERT(): KEY = sinartisi, HASH = 2,    DATA = (null), SCOPE = 0
( | 299 | 10
) | 300 | 10

 | 306 | 11
VARS | 264 | 11
INTEGER | 266 | 11
var_int1 | 308 | 11
, | 302 | 11
HASHTBL_INSERT(): KEY = var_int1, HASH = 4,     DATA = (null), SCOPE = 1
var_int2 | 308 | 11
, | 302 | 11
HASHTBL_INSERT(): KEY = var_int2, HASH = 5,     DATA = (null), SCOPE = 1
var_int3 | 308 | 11
[ | 304 | 11
10 | 310 | 11
] | 305 | 11
HASHTBL_INSERT(): KEY = var_int3, HASH = 6,     DATA = (null), SCOPE = 1
; | 301 | 11
VARS | 264 | 12
CHAR | 265 | 12
var_char | 308 | 12
; | 301 | 12
HASHTBL_INSERT(): KEY = var_char, HASH = 8,     DATA = (null), SCOPE = 1
var_int1 | 308 | 13
= | 298 | 13
var_int2 | 308 | 13
* | 290 | 13
3 | 310 | 13
+ | 287 | 13
2 | 310 | 13
; | 301 | 13
RETURN | 267 | 14
0 | 310 | 14
END_FUNCTION | 260 | 15
```

Function χωρίς return:

```
FUNCTION sinartisi()
    VARS INTEGER var_int1,var_int2,var_int3[10];
    VARS CHAR var_char;
    var_int1 = var_int2 * 3  + 2;

END_FUNCTION
```

```
FUNCTION | 259 | 10
sinartisi | 308 | 10
HASHTBL_INSERT(): KEY = sinartisi, HASH = 2,    DATA = (null), SCOPE = 0
( | 299 | 10
) | 300 | 10

 | 306 | 11
VARS | 264 | 11
INTEGER | 266 | 11
var_int1 | 308 | 11
, | 302 | 11
HASHTBL_INSERT(): KEY = var_int1, HASH = 4,     DATA = (null), SCOPE = 1
var_int2 | 308 | 11
, | 302 | 11
HASHTBL_INSERT(): KEY = var_int2, HASH = 5,     DATA = (null), SCOPE = 1
var_int3 | 308 | 11
[ | 304 | 11
10 | 310 | 11
] | 305 | 11
HASHTBL_INSERT(): KEY = var_int3, HASH = 6,     DATA = (null), SCOPE = 1
; | 301 | 11
VARS | 264 | 12
CHAR | 265 | 12
var_char | 308 | 12
; | 301 | 12
HASHTBL_INSERT(): KEY = var_char, HASH = 8,     DATA = (null), SCOPE = 1
var_int1 | 308 | 13
= | 298 | 13
var_int2 | 308 | 13
* | 290 | 13
3 | 310 | 13
+ | 287 | 13
2 | 310 | 13
; | 301 | 13
END_FUNCTION | 260 | 15

There is an error: syntax error, unexpected endfunction
Line: 15
Token: END_FUNCTION
```

Function χωρίς end_function:

```
FUNCTION sinartisi()
    VARS INTEGER var_int1,var_int2,var_int3[10];
    VARS CHAR var_char;
    var_int1 = var_int2 * 3  + 2;
    RETURN 0
```

```
FUNCTION | 259 | 10
sinartisi | 308 | 10
HASHTBL_INSERT(): KEY = sinartisi, HASH = 2,    DATA = (null), SCOPE = 0
( | 299 | 10
) | 300 | 10

 | 306 | 11
VARS | 264 | 11
INTEGER | 266 | 11
var_int1 | 308 | 11
, | 302 | 11
HASHTBL_INSERT(): KEY = var_int1, HASH = 4,     DATA = (null), SCOPE = 1
var_int2 | 308 | 11
, | 302 | 11
HASHTBL_INSERT(): KEY = var_int2, HASH = 5,     DATA = (null), SCOPE = 1
var_int3 | 308 | 11
[ | 304 | 11
10 | 310 | 11
] | 305 | 11
HASHTBL_INSERT(): KEY = var_int3, HASH = 6,     DATA = (null), SCOPE = 1
; | 301 | 11
VARS | 264 | 12
CHAR | 265 | 12
var_char | 308 | 12
; | 301 | 12
HASHTBL_INSERT(): KEY = var_char, HASH = 8,     DATA = (null), SCOPE = 1
var_int1 | 308 | 13
= | 298 | 13
var_int2 | 308 | 13
* | 290 | 13
3 | 310 | 13
+ | 287 | 13
2 | 310 | 13
; | 301 | 13
RETURN | 267 | 14
0 | 310 | 14
STARTMAIN | 268 | 17

There is an error: syntax error, unexpected startmain, expecting endfunction
Line: 17
Token: STARTMAIN
```

Main σωστή:

```
STARTMAIN
VARS INTEGER var1;
VARS CHAR var2;
VARS INTEGER counter;
```

```
ENDMAIN
```

```
STARTMAIN | 268 | 18
VARS | 264 | 19
INTEGER | 266 | 19
var1 | 308 | 19
; | 301 | 19
HASHTBL_INSERT(): KEY = var1, HASH = 8,         DATA = (null), SCOPE = 1
VARS | 264 | 20
CHAR | 265 | 20
var2 | 308 | 20
; | 301 | 20
HASHTBL_INSERT(): KEY = var2, HASH = 9,         DATA = (null), SCOPE = 1
VARS | 264 | 21
INTEGER | 266 | 21
counter | 308 | 21
; | 301 | 21
HASHTBL_INSERT(): KEY = counter, HASH = 8,      DATA = (null), SCOPE = 1
```

```
ENDIF | 280 | 55
```

Main χωρίς startmain:

```
17    END_FUNCTION
18
19    VARS INTEGER var1;
```

```
END_FUNCTION | 260 | 17


 | 306 | 19
VARS | 264 | 19

There is an error: syntax error, unexpected vars, expecting function or startmain
Line: 19
Token: VARS
```

Main χωρίς endmain:

```
56    ENDIF
57
```

```
ENDIF | 280 | 56

There is an error: syntax error, unexpected T_EOF
Line: 57
Token:
```

Εντολή ανάθεσης σωστή:

```
13      var_int2 = 5;
14      var_int1 = var_int2 * 3  + 2;
```

```
var_int2 | 308 | 13
= | 298 | 13
5 | 310 | 13
; | 301 | 13
var_int1 | 308 | 14
= | 298 | 14
var_int2 | 308 | 14
* | 290 | 14
3 | 310 | 14
+ | 287 | 14
2 | 310 | 14
; | 301 | 14
```

Εντολή ανάθεσης λάθος:

```
20    VARS var1;
```

```
VARS | 264 | 20
var1 | 308 | 20

There is an error: syntax error, unexpected id, expecting char or int
Line: 20
Token: var1
```

While σωστή:

```
WHILE(var1==5)
    var1 = var1;
ENDWHILE
```

```
WHILE | 270 | 46
( | 299 | 46
var1 | 308 | 46
== | 294 | 46
5 | 310 | 46
) | 300 | 46
var1 | 308 | 47
= | 298 | 47
var1 | 308 | 47
; | 301 | 47
ENDWHILE | 271 | 48
```

While λάθος:

```
WHILE(var1=5)
    var1 = var1;
ENDWHILE
```

```
WHILE | 270 | 47
( | 299 | 47
var1 | 308 | 47
= | 298 | 47

There is an error: syntax error, unexpected assign
Line: 47
Token: =
```

For σωστή:

```
FOR counter:=1 TO 100 STEP 2

ENDFOR
```

```
FOR | 272 | 32
counter | 308 | 32
: | 303 | 32
= | 298 | 32
1 | 310 | 32
TO | 273 | 32
100 | 310 | 32
STEP | 274 | 32
2 | 310 | 32
ENDFOR | 275 | 34
```

For λάθος:

```
FOR counter:=1 TO STEP 2
    SWITCH(var1)
    CASE(1):
        sinartisi((10 * var1),var2)
    CASE(2):
        var1 = var1;
        BREAK;
    DEFAULT:
        var1 = var1;
    ENDSWITCH
ENDFOR
```

```
FOR | 272 | 33
counter | 308 | 33
: | 303 | 33
= | 298 | 33
1 | 310 | 33
TO | 273 | 33
STEP | 274 | 33

There is an error: syntax error, unexpected step, expecting number
Line: 33
Token: STEP
```

If σωστή:

```
25    IF(var1==5) THEN
26            var1 = var1 + 12;
27        ELSEIF(var1>6)
28            var1 = var1 + 2;
29        ELSE
30            var1 = var1 + 3;
31    ENDIF
```

```
IF | 280 | 24
( | 299 | 24
var1 | 308 | 24
== | 294 | 24
5 | 310 | 24
) | 300 | 24
THEN | 277 | 24
var1 | 308 | 25
= | 298 | 25
var1 | 308 | 25
+ | 287 | 25
12 | 310 | 25
; | 301 | 25
ELSEIF | 279 | 26
( | 299 | 26
var1 | 308 | 26
> | 293 | 26
6 | 310 | 26
) | 300 | 26
var1 | 308 | 27
= | 298 | 27
var1 | 308 | 27
+ | 287 | 27
2 | 310 | 27
; | 301 | 27
ELSE | 278 | 28
var1 | 308 | 29
= | 298 | 29
var1 | 308 | 29
+ | 287 | 29
3 | 310 | 29
; | 301 | 29
ENDIF | 280 | 30
```

If λάθος:

```
IF(var1==5) THEN
        var1 = var1 + 12;
    ELSEIF(arg1>6)
        var1 = var1 + 2;
    ELSE
        var1 = var1 + 3;
ENDMAIN
```

```
IF | 280 | 50
( | 299 | 50
var1 | 308 | 50
== | 294 | 50
5 | 310 | 50
) | 300 | 50
THEN | 277 | 50
var1 | 308 | 51
= | 298 | 51
var1 | 308 | 51
+ | 287 | 51
12 | 310 | 51
; | 301 | 51
ELSEIF | 279 | 52
( | 299 | 52
arg1 | 308 | 52
> | 293 | 52
6 | 310 | 52
) | 300 | 52
var1 | 308 | 53
= | 298 | 53
var1 | 308 | 53
+ | 287 | 53
2 | 310 | 53
; | 301 | 53
ELSE | 278 | 54
var1 | 308 | 55
= | 298 | 55
var1 | 308 | 55
+ | 287 | 55
3 | 310 | 55
; | 301 | 55
ENDMAIN | 269 | 56

There is an error: syntax error, unexpected endmain, expecting else or elseif or endif
Line: 56
Token: ENDMAIN
```

Switch σωστή:

```
SWITCH(var1)
CASE(1):
    sinartisi((10 * var1),var2)
CASE(2):
    var1 = var1;
    BREAK;
DEFAULT:
    var1 = var1;
ENDSWITCH
```

```
SWITCH | 281 | 33
( | 299 | 33
var1 | 308 | 33
) | 300 | 33
CASE | 282 | 34
( | 299 | 34
1 | 310 | 34
) | 300 | 34
: | 303 | 34
sinartisi | 308 | 35
( | 299 | 35
( | 299 | 35
10 | 310 | 35
* | 290 | 35
var1 | 308 | 35
) | 300 | 35
, | 302 | 35
var2 | 308 | 35
) | 300 | 35
CASE | 282 | 36
( | 299 | 36
2 | 310 | 36
) | 300 | 36
: | 303 | 36
var1 | 308 | 37
= | 298 | 37
var1 | 308 | 37
; | 301 | 37
BREAK | 286 | 38
; | 301 | 38
DEFAULT | 283 | 39
: | 303 | 39
var1 | 308 | 40
= | 298 | 40
var1 | 308 | 40
; | 301 | 40
ENDSWITCH | 284 | 41
```

Switch σωστή χωρίς default:

```
    SWITCH(var1)
    CASE(1):
        sinartisi((10 * var1),var2)
    CASE(2):
        var1 = var1;
        BREAK;

    ENDSWITCH
```

```
SWITCH | 281 | 35
( | 299 | 35
var1 | 308 | 35
) | 300 | 35
CASE | 282 | 36
( | 299 | 36
1 | 310 | 36
) | 300 | 36
: | 303 | 36
sinartisi | 308 | 37
( | 299 | 37
( | 299 | 37
10 | 310 | 37
* | 290 | 37
var1 | 308 | 37
) | 300 | 37
, | 302 | 37
var2 | 308 | 37
) | 300 | 37
CASE | 282 | 38
( | 299 | 38
2 | 310 | 38
) | 300 | 38
: | 303 | 38
var1 | 308 | 39
= | 298 | 39
var1 | 308 | 39
; | 301 | 39
BREAK | 286 | 40
; | 301 | 40
ENDSWITCH | 284 | 42
```

Switch λάθος:

```
SWITCH(var1)
CASE:
    sinartisi((10 * var1),var2)
CASE(2):
    var1 = var1;
    BREAK;
DEFAULT:
    var1 = var1;
ENDSWITCH
```

```
SWITCH | 281 | 34
( | 299 | 34
var1 | 308 | 34
) | 300 | 34
CASE | 282 | 35
: | 303 | 35

There is an error: syntax error, unexpected colon, expecting left parenthesis
Line: 35
Token: :
```

Print σωστή:

```
PRINT("auto einai ena string),[var1]);
PRINT("auto einai ena allo string");
```

```
PRINT | 285 | 22
( | 299 | 22
String found
auto einai ena string
, | 302 | 22
[ | 304 | 22
var1 | 308 | 22
] | 305 | 22
) | 300 | 22
; | 301 | 22
HASHTBL_INSERT(): KEY = auto einai ena string, HASH = 6,        DATA = (null), SCOPE = 1
PRINT | 285 | 23
( | 299 | 23
String found
auto einai ena allo string
) | 300 | 23
; | 301 | 23
```

Print λάθος:

```
PRINT("auto einai ena string"[var1]);
```

```
PRINT | 285 | 22
( | 299 | 22
String found
auto einai ena string
[ | 304 | 22

There is an error: syntax error, unexpected left bracket, expecting right parenthesis or comma
Line: 22
Token: [
```

Break σωστή:

```
BREAK;
```

```
BREAK | 286 | 38
; | 301 | 38
```

Break λάθος:

```
BREAK
```

```
BREAK | 286 | 40
DEFAULT | 283 | 41

There is an error: syntax error, unexpected default, expecting semicolon
Line: 41
Token: DEFAULT
```

Σχόλιο μιας γραμμής σωστό:

```
%Auto einai ena single line comment
```

```
Comment found
Comment ended.
```

Σχόλιο μιας γραμμής λάθος:

```
Auto einai ena single line comment
```

```
Auto | 308 | 31
einai | 308 | 31

There is an error: syntax error, unexpected id, expecting assign or left parenthesis
Line: 31
Token: einai
```

Σχόλιο πολλών γραμμών σωστό:

```
/* Auto
einai ena
multi-line comment*/
```

Σχόλιο πολλών γραμμών λάθος:

```
/* Auto
einai ena
multi-line comment/
```

```
Multi-Lined Comment found

There is an error: syntax error, unexpected T_EOF
Line: 59
Token:
```

Σχόλια - Παραδοχές:

Μέθοδος και στρατηγική της ομάδας

Ο συγκεκριμένος κώδικας υλοποιήθηκε στην πλατφόρμα του Visual Studio Code, ένα περιβάλλον ανάπτυξης κώδικα. Χρησιμοποιήσαμε τη διανομή Ubuntu για να κατεβάσουμε τα απαραίτητα αρχεία των bison και flex ενώ μέσω της επέκτασης WSL του VSC(Visual Studio Code) αναπτύξαμε τον μεταγλωττιστή μας σε περιβάλλον Linux. Επιπλέον είχαμε στην διάθεσή μας μία ακόμα χρήσιμη επέκταση LiveShare με την οποία επεξεργαζόμασταν ταυτόχρονα, όλα τα μέλη της ομάδας, τα ίδια αρχεία ενώ ήμασταν σε συνεχή επικοινωνία μέσω της πλατφόρμας Discord.Με οδηγούς τις διαλέξεις, τα εγχειρίδια (των bison και flex), το διαδίκτυο και την προσωπική εμπειρία στον προγραμματισμό του κάθε μέλους αναπτύξαμε το παραπάνω αποτέλεσμα.

Παραδοχές

FLEX:

Όλες οι δεσμευμένες λέξεις της γλώσσας είναι με κεφαλαίους χαρακτήρες όπως ζητείται

Τα κενά και οι ειδικοί χαρακτήρες τα οποία επεξεργάζεται ο ο λεκτικός αναλυτής τα αναγνωρίζει, αλλά επιλέγει να τα αγνοεί

Η αλλαγή γραμμής αναγνωρίζεται και καταγράφεται σε σημεία που απαιτεί η εργασία, ενώ αγνοείται στα σημεία στα οποία δεν διευκρινίζεται η απαραίτητη αλλαγή γραμμής(με χρήση κατάλληλου flag στο flex κατα την ανάγνωση των tokens)

Όσα σύμβολα και χαρακτήρες ο μεταγλωττιστής δεν αναγνωρίζει τα καταγράφει ως άγνωστα(Unknown)

Για την αναγνώριση των Strings και Comments(Single-line, Multi-line) ο μεταγλωττιστής τα αναγνωρίζει χάρη στους υπο-αναλυτές.
Συγκεκριμένα τα Strings καταγράφονται και αποθηκεύονται χάρη στις συναρτήσεις strcat(), η οποία ουσιαστικά εκτέλει “push-forward” για οποιονδήποτε χαρακτήρα ακολουθει το σύμβολο “ και συνολικά τους αποθηκεύει σε έναν πίνακα(ss) ο οποίος με την σειρά του αντιγράφεται ως τυπος char *strval στο union του bison χάρη στην συνάρτηση strdup() (οι παραπάνω δύο συναρτήσεις περιέχονται στην βιβλιοθήκη “string.h”)

BISON:

Χρησιμοποιήσαμε την επιλογή “error-verbose” για την καλύτερη αναφορά σφαλμάτων ή οποία μας βοήθησε κατα το debugging ενώ ταυτοχρονα ζητείται στην εργασία

Επιπλέον άλλες επιλογές που προσθέσαμε είναι οι:
%nonassoc LOWER_THAN_ELSE
%nonassoc T_ELSE
οι οποίες χρησιμοποιήθηκαν για την επίλυση ενός σφάλματος ονόματι “Dangling else”

Για την αποθήκευση των δεδομένων επιλέξαμε την χρήση ενος hashtable με το οποίο ρυθμίζαμε ταυτόχρονα και την εμβέλεια κάθε δεδομένου όπως μας ζητήθηκε
Συγκεκριμένα κατά την διεκπεραίωση της γραμματικής απο τον parser χρησιμοποιούμε συναρτήσεις απο το εκάστοτε header file με σκοπο την

εισαγωγή των δεδομένων στο hashtable και την αναζήτηση αυτών έτσι ώστε να ρυθμίζουμε την ορθή δομή του αρχείου(input) που θα διαβάσει ο μεταγλωττιστής. Παράλληλα, αυξομειώνουμε την εμβέλεια(scope) ανάλογα με την δομή της γλώσσας προγραμματισμού που μας ζητειται να υλοποιήσουμε.

Τέλος, ορίζουμε κάποιες συναρτήσεις (yyerror, typedef_id_match) για την διευκόλυνση μας κατά το debugging και το έλεγχο στην δομή της δήλωσης μεταβλητών συγκεκριμένα

ΣΧΟΛΙΑ:

Αρχικά, η εκκίνηση της εργασιας μας δυσκόλεψε ιδιαίτερα, καθώς δεν ξέραμε από που να ξεκινήσουμε. Προσπαθήσαμε, να σχεδιάσουμε πρώτα την BNF γραμματική, όμως αν και μας βοήθησε να ξεκινήσουμε, συνειδητοποιήσαμε ότι είχε πολλά λάθη, όποτε καταλήξαμε στο να αλλάζουμε συνεχώς το που εστιάζαμε μεταξύ λεκτικού/συντακτικού αναλυτή και BNF, το οποίο οφειλόταν στην έλλειψη εμπειρίας μας για το πως πρέπει να σχεδιαστούν αυτοί οι αναλυτές.

Κατά την διάρκεια σύνταξης του κώδικα αντιμετωπίσαμε προβλήματα κυρίως στον σχεδιασμό της γραμματικής λόγω της αλλαγής γραμμής και του %empty, τα οποία μας οδηγούσαν σε αρκετές συγκρούσεις(conflicts). Ωστόσο, με αρκετή προσπάθεια καταφέραμε να τα μειώσουμε και να τα εξαλείψουμε.

Επιπρόσθετα, βρεθήκαμε αντιμέτωποι με το πρόβλημα της αποθήκευσης δεδομένων καθώς το yytext το οποίο διάβαζε τα “tokens”στην γραμματική δεν θεωρείται αξιόπιστο αν εφαρμοστεί στο bison .Έτσι επιλέξαμε να χρησιμοποιήσουμε hashtables τα οποία μας διευκόλυναν ιδιαίτερα αφού “διαβάζαμε” καθε token ως ξεχωριστό κλειδί.Θα μπορούσαμε να χρησιμοποιήσουμε και ενα global array αποθηκεύοντας το δυναμικά στην μνήμη ή κλάσεις(αν το bison υποστηριζε οντοκεντρίκο προγραμματισμό) αλλά καταλήξαμε στην πρώτη σκέψη μας η οποία μας βοήθησε επίσης και στην εγκαθίδρυση της εμβέλειας στην δομή της γλώσσας μας.