ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΜΑΘΗΜΑ: ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ
ΕΡΓΑΣΙΑ 2^{η} ΜΕΡΟΣ Β2 – Β3

ΤΜΗΜΑ: Β2
ΟΜΑΔΑ: 1η
ΜΕΛΗ ΟΜΑΔΑΣ:

1. Φράγκος Μαρίνος (ΠΑΔΑ – 20390255) 8^{ο} Εξάμηνο
2. Φριλίγκος Γρηγόριος (ΠΑΔΑ – 20390258) 8^{ο} Εξάμηνο
3. Βροχάρης Αντώνιος (ΠΑΔΑ – 20390030) 8^{ο} Εξάμηνο
4. Φάσσου Κοντοδημάκη Ιφιγένεια Γεωργία (ΠΑΔΑ – 20390249) 8^{ο} Εξάμηνο
5. Διάννης Ιωάννης (ΠΑΔΑ – 21390053) 6^{ο} Εξάμηνο

Περιεχόμενα

Μέρος Β2

Εισαγωγή

Σε αυτή την εργαστηριακή άσκηση καλούμαστε να φέρουμε σε μία πιο ολοκληρωμένη μορφή τον Συντακτικό Αναλυτή του μέρους Β1· αυτή την φορά σε συνεργασία με τον Λεκτικό Αναλυτή που είχαμε δημιουργήσει στο μέρος Α3. Πρακτικά κάνουμε τις κατάλληλες τροποποιήσεις στα δύο αυτά αρχεία με σκοπό να συνεργάζονται ώστε να αναγνωρίζουν διάφορα πρότυπα της γλώσσας Uni-C. Εφόσον ολοκληρωθούν οι αλλαγές θα δοκιμάσουμε την εκτέλεση του αρχείου μας με τα συντακτικά ορθά λεξήματα μέσω του terminal χειροκίνητα και θα συνεχίσουμε δίνοντας ως είσοδο το αρχείο input.txt στα οποία θα συγκρίνουμε τα αποτελέσματα της εκτέλεσης με αυτά που δίνονται στο αρχείο output.txt και σκοπός είναι να κατανοήσουμε τη λειτουργία της γεννήτριας καθώς και αν τα πρότυπα αναγνωρίζονται σωστά.

Σκοπός

Η παρούσα εργαστηριακή άσκηση έχει ως σκοπό να δείξουμε την εξοικείωση μας με τα εργαλεία flex και bison, καθώς επίσης να πειραματιστούμε ώστε αυτά τα δύο να συνεργάζονται μεταξύ τους. Έτσι θα μπορέσουμε να έχουμε έναν ολοκληρωμένο λεκτικό και συντακτικό αναλυτή ο οποίος είναι σε θέση να αναγνωρίζε προγράμματα σε ψευδογλώσσα Uni-C

Τεκμηρίωση κώδικα

Παρακάτω βλέπουμε τα αρχεία flex.l, bison.y, makefile, καθώς και τα αντίστοιχα αρχεία εισόδου και εξόδου που καλύπτουν τις ανάγκες της εργασίας.

Κώδικας Flex

```
/*
ΕΡΓΑΣΤΗΡΙΟ ΜΕΤΑΓΛΩΤΤΙΣΤΩΝ
2023-2024
ΤΜΗΜΑ Β2
ΟΜΑΔΑ 1
ΦΡΑΓΚΟΣ ΜΑΡΙΝΟΣ
ΦΡΙΛΙΓΚΟΣ ΓΡΗΓΟΡΙΟΣ
ΒΡΟΧΑΡΗΣ ΑΝΤΩΝΙΟΣ
ΦΑΣΣΟΥ ΚΟΝΤΟΔΗΜΑΚΗ ΙΦΙΓΕΝΕΙΑ ΓΕΩΡΓΙΑ
ΔΙΑΝΝΗΣ ΙΩΑΝΝΗΣ
*/
/*Το πρόγραμμα διαβάζει μόνο ένα αρχείο και στο πρώτο EOF τερματίζει*/
%option noyywrap

   /*Ορισμός των header files (#include...) και μεταβλητών. Οτιδήποτε ανάμεσα στα %{ %} μεταφέρεται όπως είναι στο .c αρχείο που θα δημιουργήσει το flex*/

%{

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "bison.tab.h"

// Αρχικοποίηση μεταβλητών για το άθροισμα των σωστών (cw) και λάθος (iw) λέξεων
int cw = 0;
int iw = 0;
//   Για   να   δίνουμε   στον   χρήστη   σωστό   output.char panic_cause_char[100];

%}
```

```
   /*Ονόματα και αντίστοιχοι ορισμοί. Μετά υπάρχει δυνατότητα για χρήση των ονομάτων (PAR_END, PAR_START...) αντί για χρήση κανονικών εκφράσεων, που συχνά είναι μακροσκελείς*/


PAR_END             \)
PAR_START           \(
BRACE_END           \}
BRACE_START         \{
BRACKET_END         \]
BRACKET_START       \[
SEMI                ;
HASH                #
TILDE               ~
NEQ                 !=
MOD                 \%
POW                 \^
DOT                 \.
COMMA               \,
COLON               \:
AMPER               \&
LOGICAL_OR          \|\|
TYPE_EQ             ==?
TYPE_DIV            \/=?
TYPE_MULTI          \*=?
TYPE_EXCLA          \!=?
TYPE_AMPER          \&\&
TYPE_LESSER         \<=?
TYPE_GREATER        \>=?
TYPE_PLUS           \+[\+=]?
TYPE_MINUS          \-[\-=]?
STR               ["]([^"\\\n]|\\.|\\\n)*["]
INT             0[xX][0-9a-fA-F]+|[1-9][0-9]*|0[0-7]+|0\b
COMMENT             \/\*(.|\n)*?\*\/|\/\/.*
IDENTIFIER          [A-Za-z_][A-Za-z0-9_]{0,32}
FLOAT             [[0-9]+((\.[0-9]+)([eE][+-]?[0-9]*)?|([eE][+-
]?[0-9]*)+)
WHITESPACE          [ \t]

%x REALLYEND
%x PREPANIC
%x PANIC
```

```
/*Για κάθε pattern ({MOD}, {POW}...) που ταιριάζει, εκτελείται
ο αντίστοιχος κώδικας στα άγκιστρα. Με το return επιτρέπεται η
επιστροφή μίας αριθμιτικής τιμής μέσα από την yylex() */


%%
{MOD}           {cw++; return MOD;                  }
{POW}           {cw++; return POW;                  }
{DOT}           {cw++; return DOT;                  }
{SEMI}          {cw++; return SEMI;                 }
{HASH}          {cw++; return HASH;                 }
{COMMA}         {cw++; return COMMA;                }
{PAR_END}       {cw++; return PAR_END;              }
{PAR_START}     {cw++; return PAR_START;            }
{BRACE_END}     {cw++; return BRACE_END;            }
{LOGICAL_OR}    {cw++; return LOGICAL_OR;           }
{BRACE_START}   {cw++; return BRACE_START;          }
{BRACKET_END}   {cw++; return BRACKET_END;          }
{BRACKET_START} {cw++; return BRACKET_START;        }
{FLOAT}         {cw++; return FLOAT;   }
{STR}        {cw++; return STR;  }
{INT}      {cw++; return INT;}
{IDENTIFIER}    {
      if        ( !strcmp(yytext,"do"        ) ) {cw++; return
KEYWORD;}
      else if ( !strcmp(yytext,"while"     ) ) {cw++; return
KEYWORD;}
      else if ( !strcmp(yytext,"break"     ) ) {cw++; return
KEYWORD;}
      else if ( !strcmp(yytext,"if"        ) ) {cw++; return
KEYWORD_IF;}
      else if ( !strcmp(yytext,"struct"    ) ) {cw++; return
KEYWORD_STR;}
      else if ( !strcmp(yytext,"for"       ) ) {cw++; return
KEYWORD_FOR;}
      else if ( !strcmp(yytext,"return"    ) ) {cw++; return
KEYWORD_RET;}
      else if ( !strcmp(yytext,"case"      ) ) {cw++; return
KEYWORD_CASE;}
      else if ( !strcmp(yytext,"else"      ) ) {cw++; return
KEYWORD_ELSE;}
```

```
      else if ( !strcmp(yytext,"func"      ) ) {cw++; return
KEYWORD_FUNC;}
      else if ( !strcmp(yytext,"void"      ) ) {cw++; return
KEYWORD_VOID;}
      else if ( !strcmp(yytext,"sizeof"    ) ) {cw++; return
KEYWORD_SIZE;}
      else if ( !strcmp(yytext,"include"   ) ) {cw++; return
KEYWORD_INCL;}
      else if ( !strcmp(yytext,"continue") ) {cw++; return
KEYWORD_CONT;}
      else if ( !strcmp(yytext,"switch"    ) ) {cw++; return
KEYWORD_SWITCH;}
      else if ( !strcmp(yytext,"int"       ) ) {cw++; return
KEYWORD_VAR_TYPE;}
      else if ( !strcmp(yytext,"char"      ) ) {cw++; return
KEYWORD_VAR_TYPE;}
      else if ( !strcmp(yytext,"long"      ) ) {cw++; return
KEYWORD_VAR_TYPE;}
      else if ( !strcmp(yytext,"short"     ) ) {cw++; return
KEYWORD_VAR_TYPE;}
      else if ( !strcmp(yytext,"float"     ) ) {cw++; return
KEYWORD_VAR_TYPE;}
      else if ( !strcmp(yytext,"const"     ) ) {cw++; return
KEYWORD_VAR_TYPE;}
      else if ( !strcmp(yytext,"double"    ) ) {cw++; return
KEYWORD_VAR_TYPE;}
      else if ( !strcmp(yytext,"scan"    ) ) {cw++; return
KEYWORD_SCAN;}
      else if ( !strcmp(yytext,"len"    ) ) {cw++; return
KEYWORD_LEN;}
      else if ( !strcmp(yytext,"cmp"    ) ) {cw++; return
KEYWORD_CMP;}
      else if ( !strcmp(yytext,"print"    ) ) {cw++; return
KEYWORD_PRINT;}
      else                                      {cw++; return
IDENTIFIER;}
}
{TYPE_EXCLA}    { if (!strcmp(yytext, "!=")) {cw++; return NEQ;
} else    { return EXCLA;  }}
{TYPE_EQ}       { if (!strcmp(yytext, "==")) {cw++; return EQQ;
} else    { return EQ;     }}
{TYPE_DIV}        { if (!strcmp(yytext, "/=")) {cw++; return
EQ_DIV;     } else    { return DIV;    }}
```

```
{TYPE_MULTI}      { if (!strcmp(yytext, "*=")) {cw++; return
EQ_MULTI;   } else    { return MULTI;  }}
{TYPE_LESSER}     { if (!strcmp(yytext, "<=")) {cw++; return
LESSER_EQ;  } else    { return LESSER; }}
{TYPE_GREATER}    { if (!strcmp(yytext, ">=")) {cw++; return
GREATER_EQ; } else    { return GREATER;}}
{TYPE_AMPER}      { if (!strcmp(yytext, "&&")) {cw++; return
LOGICAL_AND;} else    { return AMPER;  }}
{TYPE_MINUS}      { if (!strcmp(yytext, "--")) {cw++; return
MINUSMINUS; } else if (!strcmp(yytext, "-=")) { return EQ_MINUS;
} else { return MINUS;}}
{TYPE_PLUS}       { if (!strcmp(yytext, "++")) {cw++; return
PLUSPLUS;   } else if (!strcmp(yytext, "+=")) { return EQ_PLUS;
} else { return PLUS; }}
{WHITESPACE}    { }
{COMMENT}       {}
\n              {return NEWLINE; }
 /*Εδώ το flex εντοπίζει οποιονδήποτε χαρακτήρα που δεν
περιγράφεται απο τις παραπάνω κανονικές εκφράσεις.*/
<INITIAL>.  { BEGIN(PREPANIC); strcpy(panic_cause_char,yytext);
iw++; return UNKNOWN;}
<PREPANIC>.     {     BEGIN(PANIC);     printf("Unknown     word:
'%s%s",panic_cause_char,yytext);}
<PANIC>{WHITESPACE} { printf("'\n"); BEGIN(INITIAL);}
<PANIC>\n           { printf("'\n"); BEGIN(INITIAL);}
<PANIC>.            { ECHO; }
 /*Εδώ καλούμε ένα τμήμα κώδικα που μας βοηθά να δώσουμε ένα
token στον bison για να δηλώσουμε το τέλος του αρχείου, χωρίς
όμως να τερματίζεται άμεσα η εκτέλεση του bison.
 Έτσι, καταφέρνουμε να εκτελούμε την συνάρτηση exp_report() στο
bison.y, για να ανεφέρουμε τον αριθμό των σωστών και λανθασμένων
λέξεων και εκφράσεων.*/
<INITIAL><<EOF>>  { BEGIN(REALLYEND);
                    printf("Syntax Report:\nThe program counted
(%d)  words,\nOf  which  (%d)  were  correct,\nAnd  (%d)  were
incorrect.\n",cw+iw,cw, iw);
                    return EOP; }
 /*Εδώ, μετά την πάροδο των προηγούμενων, τερματίζουμε την
εκτέλεση του flex, έχουμε ήδη τυπώσει την αναφορά με την
exp_report() με το bison.*/
<REALLYEND><<EOF>> {yyterminate();}
%%
```

```
/* Δεν υπάρχει main(), καθώς δεν τρέχει αυτόνομα, είναι απλά ο λεκτικός αναλυτής, η συντακτική ανάλυση γίνεται από τον bison. */
```

Σχόλια

Στον παραπάνω κώδικα βλέπουμε το αρχείο του flex. Ο παρόν κώδικας αποτελεί παραγόμενο προϊόν του μέρους Α3. Πρακτικά στο συγκριμένο αρχείο έχουμε επέμβει προκειμένου να συμπληρώσουμε ότι χρειάζεται ώστε να μπορεί να στείλει τα κατάλληλα αναγνωριστικά στον bison. Επίσης, πλέον δεν χρησιμοποιούμε header file (+ τον ανάλογο πίνακα αναγκωριστικών) μιας και αυτό το παράγει ο bison και τον χρησιμοποιεί ο flex.

Κώδικας Bison

```
/*
ΕΡΓΑΣΤΗΡΙΟ ΜΕΤΑΓΛΩΤΤΙΣΤΩΝ
2023-2024
ΤΜΗΜΑ Β2
ΟΜΑΔΑ 1
ΦΡΑΓΚΟΣ ΜΑΡΙΝΟΣ
ΦΡΙΛΙΓΚΟΣ ΓΡΗΓΟΡΙΟΣ
ΒΡΟΧΑΡΗΣ ΑΝΤΩΝΙΟΣ
ΦΑΣΣΟΥ ΚΟΝΤΟΔΗΜΑΚΗ ΙΦΙΓΕΝΕΙΑ ΓΕΩΡΓΙΑ
ΔΙΑΝΝΗΣ ΙΩΑΝΝΗΣ
*/
%{
   /*Αρχεία header (#include...), δηλώσεις define (εδώ δεν υπάρχει κάποια), αρχικοποίηση
   μεταβλητών και συναρτήσεων που θα χρησιμοποιήσουμε στο πρόγραμμα*/
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

extern int yylex(void);
extern int yyparse(void);

extern FILE *yyin;
```

```
void yyerror(char *);
void exp_report(int,int);
void yytrue (char *);

int ce  = 0;
int ie  = 0;

int function_start_line=0;
int function_started_flag=0;

// Μεταβλητή line για την μέτρηση γραμμών
int line=1;
%}



// Ορισμός των λεκτικών μονάδων
%token EOP
    UNKNOWN
     DOT
     SEMI
     HASH
     COLON
     COMMA
     FLOAT
     DOUBLE
     STR
     NEWLINE
     KEYWORD
     INT
     IDENTIFIER
     KEYWORD_IF
     AMPER EXCLA
     KEYWORD_RET
     KEYWORD_FOR
     KEYWORD_STR
     KEYWORD_ELSE
     KEYWORD_SIZE
     KEYWORD_CONT
     KEYWORD_CASE
     KEYWORD_INCL
     KEYWORD_FUNC
     KEYWORD_VOID
     KEYWORD_SWITCH
```

```
     KEYWORD_VAR_TYPE
     KEYWORD_SCAN
     KEYWORD_LEN
     KEYWORD_CMP
     KEYWORD_PRINT
     PAR_START PAR_END
     BRACE_START BRACE_END
     LOGICAL_OR LOGICAL_AND
     BRACKET_START BRACKET_END
     GREATER LESSER GREATER_EQ LESSER_EQ
     EQQ EQ NEQ EQ_MULTI EQ_DIV EQ_PLUS EQ_MINUS
     PLUS PLUSPLUS MINUS MINUSMINUS DIV MOD MULTI POW

//Προτεραιότητα των tokens (από την χαμηλώτερη στην υψηλότερη)
%left  POW
%left  PLUS MINUS
%left  DIV MULTI

%%
   /*Ορισμός των γραμματικών κανόνων. Όταν αντιστοιχίζεται ένας
κανόνας με
   τα δεδομένα που δίνει ο χρήστης ως είσοδο, εκτελείται ο
κώδικας στα άγκιστρα {}*/
program:
    program valid
    |
    ;


/* Εδώ ορίζεται το τι μπορεί να είναι κομμάτι μίας έκφρασης.
   Ένας χαρακτήρας (str) ή ένας αριθμός (ακέραιος ή πραγματικός)
*/
expr_part:
      FLOAT
    | STR
    | DOUBLE
    | KEYWORD
    | INT
    | IDENTIFIER
    | UNKNOWN { printf("(X) \tLine:  %d \t",line); }
    ;

// Οι τελεστές
```

```
operator:
      EQ
    | EQQ
    | NEQ
    | DIV
    | POW
    | PLUS
    | MINUS
    | MULTI
    | EQ_DIV
    | EQ_PLUS
    | EQ_MULTI
    | EQ_MINUS
    ;

in_de_crement_operator:
    | MINUSMINUS
    | PLUSPLUS
    ;

// Οι εκφράσεις υπο επεξεργασία
expr_proc:
      expr_part operator expr_part EQ expr_part
    | expr_part operator expr_part
    | expr_part operator expr_proc
    | expr_part in_de_crement_operator
    | in_de_crement_operator expr_part
    ;

// Το κύριο κομμάτι του κώδικα, ένας αριθμός συντακτικά σωστών
εκφράσεων.
body:
    body valid
    | valid
    |
    ;

elements:
    expr_part COMMA elements
    | expr_part
    ;
```

```
// Παρακάτω ορίζεται τι μπορεί να βρίσκεται μέσα σε αγγύλες (brackets) και τι μέσα σε άγκυστρο (braces)
in_brack:
    BRACKET_START elements BRACKET_END

in_brace:
    BRACE_START body BRACE_END

struct:
      KEYWORD_STR IDENTIFIER in_brace
    | KEYWORD_STR IDENTIFIER NEWLINE in_brace
    ;

loops:
    for_grammar

// Ποιά μπορούν να είναι ορίσματα μιας συνάρτησης και πώς ορίζεται μια συνάρτηση
arguments:
    expr_part
    | expr_part COMMA arr_help
    | KEYWORD_VOID
    |
    ;

arr   : BRACKET_START arr_help BRACKET_END SEMI
      | BRACKET_START arr_help BRACKET_END SEMI
      | BRACKET_START arr_help BRACKET_END SEMI
      | IDENTIFIER BRACKET_START INT BRACKET_END SEMI
      ;

arr_help: expr_part
            | expr_part COMMA expr_part
            ;

func_par:
      KEYWORD_FUNC IDENTIFIER PAR_START arguments PAR_END {ce++; yytrue("arguments"); }
    | KEYWORD_FUNC IDENTIFIER PAR_START expr_part PAR_END {ce++; yytrue("argument"); }
    ;
```

```
// Ορισμός μιας μεταβλητής (τύπος μεταβλητής και αν είναι για παράδειγμα πίνακας ή όχι)
declaration:
      KEYWORD_VAR_TYPE IDENTIFIER
    | KEYWORD_VAR_TYPE IDENTIFIER EQ expr_proc
    | KEYWORD_VAR_TYPE IDENTIFIER in_brack EQ expr_proc
    |  KEYWORD_VAR_TYPE  IDENTIFIER  in_brack  EQ  BRACE_START elements BRACE_END
    | KEYWORD_VAR_TYPE IDENTIFIER in_brack
    | KEYWORD_VAR_TYPE IDENTIFIER EQ sizeof
    ;

// Διαδικασία ανάθεσης τιμής σε μεταβλητή
assignment:
    IDENTIFIER EQ expr_proc

// Ο κανόνας για τις επιστροφές συναρτήσεων (με το return) και για τα includes
return:
    KEYWORD_RET expr_proc
    |
    |
    ;

include:
    HASH KEYWORD_INCL LESSER IDENTIFIER DOT IDENTIFIER GREATER
    | HASH KEYWORD_INCL STR
    ;
scan:
    KEYWORD_SCAN PAR_START expr_part PAR_END SEMI
    |
    ;
len:
    KEYWORD_LEN PAR_START expr_part PAR_END SEMI
    |
    ;
cmp:
    KEYWORD_CMP PAR_START expr_part COMMA expr_part PAR_END SEMI
    |
    ;
cases:
    KEYWORD_CASE COLON valid NEWLINE cases
    | KEYWORD_CASE COLON valid NEWLINE
```

```
case_grammar:
    KEYWORD_SWITCH PAR_START expr_proc PAR_END BRACE_START cases
BRACE_END
    |  KEYWORD_SWITCH  PAR_START  expr_part  PAR_END  BRACE_START
cases BRACE_END
    ;

else_grammar:
    KEYWORD_ELSE in_brace

if_grammar:
      KEYWORD_IF PAR_START expr_proc PAR_END in_brace
    | KEYWORD_IF PAR_START expr_proc PAR_END expr_proc NEWLINE
    ;

for_grammar:
      KEYWORD_FOR PAR_START for_args PAR_END in_brace
    | KEYWORD_FOR PAR_START for_args PAR_END expr_proc NEWLINE
    ;

for_args:
      expr_proc SEMI expr_proc SEMI expr_proc
    | SEMI expr_proc SEMI expr_proc
    | expr_proc SEMI SEMI expr_proc
    | expr_proc SEMI SEMI
    | SEMI expr_proc SEMI
    | SEMI SEMI expr_proc
    | SEMI SEMI
    ;
print:
    KEYWORD_PRINT PAR_START STR COMMA IDENTIFIER PAR_END SEMI
    |KEYWORD_PRINT PAR_START expr_part PAR_END SEMI
    ;
// Ο κανόνας αυτός χρησιμοποιήται μαζί με το sizeof (πχ.
sizeof(smth) * 10)
// Με το "* 10" να είναι το "half_expr"
half_expr:
      operator IDENTIFIER
    | operator INT
    | operator DOUBLE
    | operator FLOAT
    ;
```

```
// Ο κανόνας για το sizeof
sizeof:
      KEYWORD_SIZE PAR_START KEYWORD_VAR_TYPE PAR_END
    | KEYWORD_SIZE PAR_START KEYWORD_VAR_TYPE PAR_END half_expr
    ;

// Όλοι οι συντακτικοί κανόνες των if/else/case
conditionals:
      if_grammar
    | else_grammar
    | case_grammar
    ;

// Αυτά που θεωρούνται συντακτικά σωστά
valid:
     return      SEMI { ce++; yytrue("return");}
   | sizeof      SEMI { ce++; yytrue("sizeof");}
   | include     SEMI { ce++; yytrue("include");}
   | expr_proc   SEMI { ce++; yytrue("expression");}
   | assignment  SEMI { ce++; yytrue("assignment");}
   | declaration SEMI { ce++; yytrue("declaration");}
   | loops            { ce++; yytrue("loop clause");}
   | in_brace         { ce++;
                        if( function_started_flag)
                        {
                            function_started_flag=0;
                            if (line == function_start_line)
                            {
                               printf("Line:  %d \tFunction is
correct.\n",function_start_line);
                            }     else     if     (line     >=
function_start_line) {
                               printf("Lines:   %d-%d\tFunction
is correct.\n",function_start_line, line);
                            }
                        } else {
                            function_started_flag=1;
                            function_start_line=line;
                        }
                      }
   | struct SEMI      { ce++; yytrue("struct");}
   | func_par         { ce++; yytrue("function declaration");}
```

```
   | conditionals     { ce++; yytrue("conditional clause");  }
   | scan             { ce++; yytrue("scan");  }
   | len              { ce++; yytrue("len");  }
   | cmp              { ce++; yytrue("cmp");  }
   | arr              { ce++; yytrue("array"); }
   | print            { ce++; yytrue("print statement"); }
   | NEWLINE          { line++; }
   | EOP              { exp_report(ce,ie); }
   | error            { ie++;}
   ;

%%

/* Αυτή η συνάρτηση ενεργοποιείται όταν ο bison δεχτεί token EOP
και τυπώνει το πλήθος των σωστών και λάθος λέξεων και εκφράσεων*/

// (End of Parse, στο τέλος του αρχείου)
void exp_report (int ce,int ie) {
    printf("\n==================\n"
        "\nThe program counted (%d) expressions,\nOf which (%d)
were correct,\nAnd (%d) were incorrect.\n",ce+ie,ce,ie);
}

void yytrue (char * type)
{
    printf("Line:%d \tCorrect %s\n"    ,line, type);
}

void yyerror(char *s)
{
    fprintf(stderr, "(X)\tOn Line:%d \tError: %s\n",line, s);
}

//Αναγκαίες εντολές για την εκτέλεση debugging στον Bison
#ifdef YYDEBUG
  int yydebug = 1;
#endif

   /*Η συνάρτηση main, που σηματοδοτεί την εκκίνηση του
προγράμματος.
   Εδώ απλά καλεί την συνάρτηση yyparse του Bison και ξεκινάει
η συντακτική
   ανάλυση*/
```

```
int main(int argc, char* argv[])
{
    FILE *fp;
    if(argc<2)
    {
        printf("No arguments found, defaulting to
input.txt...\n");
        fp = fopen("input.txt","r");
    }
    if(argc==2)
         fp = fopen(argv[1],"r");
    yyin = fp;
    printf("\nBeginning analysis:\n");
    yyparse();
}
```

Σχόλια

Στον παραπάνω κώδικα βλέπουμε το αρχείο του bison. Ο παρόν κώδικας αποτελεί παραγόμενο προϊόν του μέρους Β3. Πρακτικά στο συγκριμένο αρχείο έχουμε επέμβει προκειμένου να αφαιρέσουμε τον ανεξάρτητο λεκτικό αναλυτή, να δημιουργήσουμε τα πρότυπα αναγνώρισης και να δημιουργήσουμε τους κατάλληλους διαύλους για μία επιτυχημένη συνεργασία με τον flex. Βέβαια είναι αναγκαίο να τονίσουμε ότι μέρος κάποιων στοιχείων δεν καταφέραμε να τα υλοποιήσουμε στο βέλτιστο. Επομένως ως ομάδα δηλώνουμε ρητά ότι το παρόν πρόγραμμα μπορεί να εκτελεστεί και να παραχθεί εκτελέσιμο όμως δεν είναι σε θέση να καλύψει όλες τις υπο-προϋποθέσεις της άσκηση όπως εμείς οι ίδιοι επιθυμούσαμε.

Κώδικας Makefile

Στο αρχείο παραθέτουμε όλες τι αυτοματοποιημένες εντολές που χρειαζόμαστε για την παραγωγή του header file, του κώδικα C, του εκτελέσιμου και των αρχείων εξόδου.

```
all:
     bison -d bison.y
     flex flex.l
     gcc -o project bison.tab.c lex.yy.c
     ./project
clean:
     rm project bison.tab.c lex.yy.c bison.tab.h
```

Δοκιμαστικές εκτελέσεις

Για τις δοκιμαστικές εκτελέσεις χρησιμοποιήθηκε ένα εκτενές αρχείο έτσι ώστε να έχουμε ολοκληρωμένη εικόνα της λειτουργίας του κώδικα που αναπτύξαμε χρησιμοποιώντας ως είσοδο. Να τονίσουμε ότι στο συγκεκριμένο αρχείο όλα είναι ορθά και δείχνουν την επιτυχία του προγράμματος μας με όποια παραδοχή έχει γίνει παραπάνω.

Αρχείο Εισόδου - Input.txt

```
3 + 2;
15 + 25 + 35;
testvar + 1;
var1 + _var2;

func main()
{

int a;
int b;
myArr[3];
print("I can print %d",a);
print(myAdd);
scan(a);
scan(b);
myAdd(a,b);
while(a<b)
{
     a++;
     printf(a);
};
return;
}

func myAdd(int num1, int num2)
{
int sum;
sum = a + b;
len("SuperString");
cmp("a", "b");
```

```
print("Sum is: ", sum)
}
```

Αρχείο εξόδου- terminal

Παρακάτω βλέπουμε το αρχείο εξόδου που παράχθηκε από τον κώδικα μας.

```
Beginning analysis:
Line:1  Correct expression
Line:2  Correct expression
Line:3  Correct expression
Line:4  Correct expression
Line:6  Correct arguments
Line:6  Correct function declaration
Line:9  Correct declaration
Line:10         Correct declaration
Line:11         Correct array
Line:12         Correct print statement
Line:13         Correct print statement
Line:14         Correct scan
Line:15         Correct scan
(X)     On Line:16      Error: syntax error
Line:16         Correct return
(X)     On Line:17      Error: syntax error
Line:19         Correct expression
(X)     On Line:20      Error: syntax error
Line:20         Correct return
Line:21         Correct return
(X)     On Line:22      Error: syntax error
Lines: 21-23    Function is correct.
(X)     On Line:25      Error: syntax error
Line:27         Correct declaration
Line:28         Correct assignment
Line:29         Correct len
Line:30         Correct cmp
(X)     On Line:31      Error: syntax error
Syntax Report:
The program counted (113) words,
Of which (113) were correct,
And (0) were incorrect.

==================
```

```
The program counted (47) expressions,
Of which (24) were correct,
And (23) were incorrect.
```

Σχολιασμός αρχείου εξόδου

Ο κώδικας κατά βάση είναι αποδοτικός και αναγνωρίζει αρκετές εκφράσεις της γλώσσας Uni-C. Παρ' όλα αυτά δεν είναι απόλυτα αποτελεσματικός καθώς έχει πολλές ελλείψεις τις οποίες δεν καταφέραμε να αντιμετωπίσουμε. Κρίνοντας από το αποτέλεσμα του τερματικού συγκεντρώσαμε τις παρακάτω παρατηρήσεις:

1. Στην γραμμή 16 του αρχείου εισόδου γίνεται κλήση της συνάρτησης myAdd(a, b), η οποία ορίζεται παρακάτω αλλά δεν αναγνωρίζεται από τον compiler.
2. Στη γραμμή 17 η έκφραση while(a<b) δεν αναγνωρίζεται.
3. Στη γραμμή 20 η έκφραση printf(a); δεν αναγνωρίζεται.
4. Στη γραμμή 22 η εντολή return δεν αναγνωρίζεται.
5. Στη γραμμή 25 η δήλωση της συνάρτησης myAdd δεν αναγνωρίζεται.
6. Στη γραμμή 31 η έκφραση print("Sum is: ", sum) δεν αναγνωρίζεται.

Συμπεράσματα

Στην παρούσα εργαστηριακή άσκηση προσπαθήσαμε να δημιουργήσουμε μία ολοκληρωμένη μηχανή λεκτικής και συντακτικής ανάλυσης με την βοήθεια των δύο προηγούμενων εργασιών, Α3 και Β1, και πιο συγκεκριμένα με τα εργαλεία flex και bison. Καταφέραμε να καλύψουμε όσο το δυνατόν περισσότερες προϋποθέσεις της ψευδογλώσσας Uni-C έχοντας όμως μερικές μικρές αστοχίες σε κάποιες από τις υποπεριπτώσεις κατά την προσπάθειά μας. Παρόλα αυτά είμαστε ικανοποιημένοι ως ένα βαθμό μιας και το τελικό προϊόν είναι σε θέση να παράξει εκτελέσιμο αρχείο και να μπορέσει να αναγνωρίσει πληθώρα λεξημάτων.

Μέρος Β3

Εισαγωγή

Στο τελευταίο μέρος της εργασίας καλούμαστε να ολοκληρώσουμε την υλοποίησή για την καταμέτρηση των λανθασμένων εκφράσεων χρησιμοποιώντας πλέον την μέθοδο του πανικού στον Συντακτικό Αναλυτή μας. Ο συγκεκριμένος κώδικας αφορά τον εμπλουτισμό των γραμματικών κανόνων που έχουμε δημιουργήσει στον κώδικα bison με κανόνες αναγνώρισης συνηθισμένων συντακτικών προειδοποιητικών λαθών ώστε να μπορεί γίνεται ανάνηψη από τα σφάλματα και η συντακτική ανάλυση να συνεχίζει την αναγνώριση στην αρχή της επόμενης συμβολοσειράς εισόδου, εμφανίζοντας στο αρχείο εξόδου τα κατάλληλα μηνύματα λαθών και αυξάνοντας τους αντίστοιχους μετρητές. Τέλος, χρειάζεται αντίστοιχα και ο κώδικας flex να εμπλουτιστεί με επιπλέον κανόνες εντοπισμού προειδοποιητικών λαθών και αντιμετώπισής τους, με κατάλληλη δημιουργία αντίστοιχων κανονικών εκφράσεων, εμφανίζοντας στο αρχείο εξόδου τα κατάλληλα μηνύματα και αυξάνοντας τους αντίστοιχους μετρητές λαθών.

Σκοπός

Σκοπός της παρούσας εργασίας είναι να ολοκληρώσουμε το project μας καθώς και να έχουμε μία ολοκληρωμένη εικόνα για τον συνδυασμό του λεκτικού και του συντακτικού αναλυτή. Πιο συγκεκριμένα καλούμαστε να βελτιώσουμε και να προσθέσουμε κώδικα προκειμένου να γίνει ορθή διαχείριση σφαλμάτων και σημείων προσοχής. Έτσι θα έχουμε ένα πλήρως λειτουργικό πρόγραμμα το οποίο θα είναι σε θέση να λάβει μεγάλα κομμάτια εισόδου και να μας υποδείξει το κατά πόσο η είσοδός μας είναι σωστή καθώς και πόσα ακριβώς σφάλματα έχει. Ακόμα καλούμαστε να κάνουμε μία γενική ανασκόπηση στο project προκειμένου να αντιληφθούμε την πορεία του καθώς και να εντοπίσουμε πιθανά λάθη καθώς πράγματα τα οποία θα είμαστε σε θέση να τα επεξεργαστούμε με μία διαφορετική οπτική γωνία εφόσον το project φτάνει στο τέλος του και έχουμε αποκτήσει μία πιο ολοκληρωμένη και σφαιρική γνώση γύρω από το αντικείμενο.

Σημείωση

Από τα τρία ερωτήματα του μέρους Β έχει υλοποιηθεί το πρώτο όπου αν μία έκφραση δεν αναγνωρίζεται ως έγκυρη καλείται η ρουτίνα διαχείρισης λαθών syntax error. Επίσης, ο μετρητής των σημαντικών λαθών (fatal errors) αυξάνεται κατά ένα κάθε φορά που καλείται η syntax error. Η ανάλυση συνεχίζεται στην αρχή της επόμενης συμβολοσειράς εισόδου. (Κάποιες έγκυρες εκφράσεις αναγνωρίζονται ως λανθασμένες). Δεν έχει δημιουργηθεί αρχείο εξόδου και τα αποτελέσματα εμφανίζονται στον τερματικό. Τέλος δεν έχουν ενταχθεί στον κώδικα κατάλληλα μηνύματα λαθών για τις λανθασμένες εκφράσεις παρά μόνο το syntax error, παρ όλα αυτά έχουν προστεθεί μετρητές που εμφανίζουν τον συνολικό αριθμό γραμμών και τον αριθμό σωστών και λανθασμένων εκφράσεων. Οι προσθήκες αυτές έχουν πραγματοποιηθεί από το μέρος Β2 οπότε οι κώδικες παραμένουν ως έχουν.

Αρχείο εισόδου - input.txt

```
//Δήλωση μεταβλητών
int a;
int b;

//Εκχώρηση τιμής σε μεταβλητή
a = -30;
b = a;

x1, x2 = 0, 1;
x1, list1 = 0, [1, 2, 3];
x1, list1, string = 0, [1, 2, 3], "HELLO";

//Δήλωση πινάκων
pin = [1,2,3];
pin1 = [1, 2, 3, 4, 5 ];
pin2 = ["a", "b", "c", "d"];

//Εκτύπωση στοιχείου πίνακα
print(pin1[0]);

//Συναρτήσεις
scan(x);
scan(MyVariable);

len([10, 20, 30, 40, 50]);
len("This is a string");
len(StringVariable);
```

```
cmp("test", "best");
cmp(str1, str2);

print("Hello World");
print(x, "=", 100);

print(cmp(str1, str2));
print(len("This is a string"));

//Δήλωση συνάρτησης
func myfunc(int varA, int varB)
{
 print("a = ", varA);
 print("b = ", varB);
}

//Κάλεσμα συνάρτησης
myfunc(10,20);
myfunc(var1,var2,var3);
myfunc("hello");

//Αριθμητικές Πράξεις Ακεραίων
1 + 2
-5 + 10
1 - 4
2 * 4
4 / 2
4 / 5
15 + 3 - 9

//Εκχώρηση αποτελέσματος αριθμητικής πράξης σε μεταβλητή
var1 = 3 + 7;



//Συγκρίσεις
5 == 52
1 < 2

//Δοκιμή εκχώρησης και σύγκρισης μεταβλητών
i = 2;
y = 3;
i < y
```

```
i > y
i == y

//Συνθήκες
if (var == var1) print("Value of expression is var1");
if (x < y) { print(x); print(y); print(z); }

//Επανάληψη while
while (i < w)
{
 print("hello");
 if (var == var1) {
     print("Hello");
 }
}

//Επανάληψη for
for (i = 0; i < w; i++)
{
 print("i = ", i);
}
```

Αρχείο εξόδου- terminal

```
Beginning analysis:
Line:1  Correct expression
Line:2  Correct expression
Line:3  Correct expression
Line:4  Correct expression
Line:6  Correct arguments
Line:6  Correct function declaration
Line:9  Correct declaration
Line:10         Correct declaration
Line:11         Correct array
Line:12         Correct print statement
Line:13         Correct print statement
Line:14         Correct scan
Line:15         Correct scan
(X)     On Line:16      Error: syntax error
Line:16         Correct return
(X)     On Line:17      Error: syntax error
Line:19         Correct expression
(X)     On Line:20      Error: syntax error
Line:20         Correct return
Line:21         Correct return
(X)     On Line:22      Error: syntax error
Lines: 21-23    Function is correct.
(X)     On Line:25      Error: syntax error
Line:27         Correct declaration
Line:28         Correct assignment
Line:29         Correct len
Line:30         Correct cmp
(X)     On Line:31      Error: syntax error
Syntax Report:
The program counted (113) words,
Of which (113) were correct,
And (0) were incorrect.

==================

The program counted (47) expressions,
Of which (24) were correct,
And (23) were incorrect.
marinosfrangos@192 Part B % make
bison -d bison.y
```

```
bison.y: conflicts: 209 shift/reduce, 125 reduce/reduce
bison.y:132.7-34:  warning:  rule  never  reduced  because  of
conflicts: expr_proc: expr_part operator expr_part
bison.y:167.5-13:  warning:  rule  never  reduced  because  of
conflicts: arguments: expr_part
bison.y:174.9-47:  warning:  rule  never  reduced  because  of
conflicts: arr: BRACKET_START arr_help BRACKET_END SEMI
bison.y:175.9-47:  warning:  rule  never  reduced  because  of
conflicts: arr: BRACKET_START arr_help BRACKET_END SEMI
bison.y:216.6: warning: rule never reduced because of conflicts:
scan: /* empty */
bison.y:220.6: warning: rule never reduced because of conflicts:
len: /* empty */
bison.y:224.6: warning: rule never reduced because of conflicts:
cmp: /* empty */
flex flex.l
gcc -o project bison.tab.c lex.yy.c
./project
No arguments found, defaulting to input.txt...

Beginning analysis:
Line:2  Correct declaration
Line:3  Correct declaration
(X)     On Line:6       Error: syntax error
Line:6  Correct expression
Line:7  Correct assignment
(X)     On Line:9       Error: syntax error
(X)     Line:  9        Unknown word: '0,'
(X)     On Line:9       Error: syntax error
Line:9  Correct expression
(X)     On Line:10      Error: syntax error
(X)     Line:  10       Unknown word: '0,'
(X)     On Line:10      Error: syntax error
(X)     On Line:10      Error: syntax error
Line:10         Correct return
(X)     On Line:11      Error: syntax error
(X)     Line:  11       Unknown word: '0,'
(X)     On Line:11      Error: syntax error
(X)     On Line:11      Error: syntax error
Line:11         Correct expression
(X)     On Line:14      Error: syntax error
(X)     On Line:14      Error: syntax error
Line:14         Correct return
```

```
(X)     On Line:15      Error: syntax error
(X)     On Line:15      Error: syntax error
Line:15         Correct return
(X)     On Line:16      Error: syntax error
(X)     On Line:16      Error: syntax error
Line:16         Correct return
(X)     On Line:19      Error: syntax error
(X)     Line:  19       Unknown word: '0]);'
Line:21         Correct scan
Line:22         Correct scan
(X)     On Line:24      Error: syntax error
(X)     On Line:24      Error: syntax error
Line:24         Correct return
Line:25         Correct len
Line:26         Correct len
Line:28         Correct cmp
Line:29         Correct cmp
Line:31         Correct print statement
(X)     On Line:32      Error: syntax error
Line:32         Correct return
(X)     On Line:34      Error: syntax error
(X)     On Line:34      Error: syntax error
Line:34         Correct return
(X)     On Line:35      Error: syntax error
(X)     On Line:35      Error: syntax error
Line:35         Correct return
(X)     On Line:38      Error: syntax error
Line:40         Correct print statement
Line:41         Correct print statement
(X)     On Line:45      Error: syntax error
Line:45         Correct return
(X)     On Line:46      Error: syntax error
Line:46         Correct return
(X)     On Line:47      Error: syntax error
Line:47         Correct return
(X)     On Line:50      Error: syntax error
(X)     On Line:51      Error: syntax error
(X)     On Line:52      Error: syntax error
(X)     On Line:53      Error: syntax error
(X)     On Line:54      Error: syntax error
(X)     On Line:55      Error: syntax error
(X)     On Line:56      Error: syntax error
Line:59         Correct assignment
```

```
(X)     On Line:63      Error: syntax error
Line:67         Correct assignment
Line:68         Correct assignment
(X)     On Line:69      Error: syntax error
(X)     On Line:71      Error: syntax error
(X)     On Line:74      Error: syntax error
Line:74         Correct print statement
(X)     On Line:75      Error: syntax error
Line:75         Correct print statement
Line:75         Correct print statement
Line:75         Correct print statement
Lines: 42-75    Function is correct.
(X)     On Line:78      Error: syntax error
Line:80         Correct print statement
Line:82         Correct print statement
Line:83         Correct conditional clause
(X)     Line:  87       Unknown word: '0;'
(X)     On Line:87      Error: syntax error
Line:87         Correct expression
(X)     On Line:87      Error: syntax error
Line:89         Correct print statement
Lines: 84-90    Function is correct.
Syntax Report:
The program counted (325) words,
Of which (320) were correct,
And (5) were incorrect.

==================

The program counted (168) expressions,
Of which (42) were correct,
And (126) were incorrect.
```

Κατανομή εργασιών

Υπεύθυνος εργασίας Β-2:	Φράγκος Μαρίνος
Δημιουργία - Τροποποίηση κώδικα bison:	Διάννης Ιωάννης, Βροχάρης Αντώνιος
Δημιουργία - Τροποποίηση κώδικα flex:	Διάννης Ιωάννης, Βροχάρης Αντώνιος
Δημιουργία αρχείου input:	Διάννης Ιωάννης, Βροχάρης Αντώνιος
Δοκιμαστικές εκτελέσεις:	Φράγκος Μαρίνος, Φριλίγκος Γρηγόριος, Φάσσου Κοντοδημάκη Ιφιγένεια Γεωργία
Σύνταξη εγγράφου:	Φάσσου Κοντοδημάκη Ιφιγένεια Γεωργία, Φράγκος Μαρίνος, Φριλίγκος Γρηγόριος