

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

Αρχές Γλωσσών Προγραμματισμού & Μεταφραστών

Εαρινό Εξάμηνο 2016



ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Ομάδα:

Κανούτος Κωνσταντίνος 5775

Κυριακού Ανδρόνικος 5806

Ντενέζος Παναγιώτης 5853

Πλούμης Θωμάς 5880

Ερώτημα 1^ο

Δίνεται η παρακάτω γραμματική σε συμβολισμό EBNF που περιγράφει, χωρίς πολλές λεπτομέρειες, τους ορισμούς συναρτήσεων της C και ένα μικρό υποσύνολο των εντολών της. (Παραλείπονται οι ορισμοί των μη τερματικών συμβόλων `simple`, `formals` και `expr`.)

```
<decl> ::= <result> <id> ( <formals> ) { <stmt>* }  
<result> ::= <type> | void  
<stmt> ::= <simple> | break ; | continue ; | return [ <expr> ] ;  
           | while ( <expr> ) { <stmt>* } | switch ( <expr> ) { <stmt>* }
```

Ο κώδικας που χρησιμοποιήθηκε για να δημιουργεί συμβολοσειρές που να πληρούν τους περιορισμούς της εκφώνησης είναι ο εξής:

```
<decl> ::= <type><id>(<formals>){<stmtA>*}|<void><id>(<formals>){<stmtB>*}  
<stmtA> ::= <simple> | <whA> | <swA> | [return <expr>;]  
<stmtB> ::= <simple> | <whB> | <swB>
```

```
<whA> ::= while(<expr>){ (break; | continue; | <stmtA>)* } <stmtA>*  
<whB> ::= while(<expr>){ (break; | continue; | <stmtB>)* } <stmtB>*
```

```
<swA> ::= switch(<expr>){ (break; | <stmtA>)* } <stmtA>*  
<swB> ::= switch(<expr>){ (break; | <stmtB>)* } <stmtB>*
```

Για την επίτευξη των διαφορετικών περιπτώσεων της επιστρεφόμενης τιμής, δεδομένου ότι δεν υπάρχει κανόνας για την `<result>`, προσθέσαμε περιορισμούς στον κανόνα της `<decl>`. Τα μη τερματικά σύμβολα ονομάστηκαν με βάση το περιεχόμενο τους και χωρίστηκαν σε A και B ανάλογα με τον τύπο επιστρεφόμενης τιμής .

Ερώτημα 2ο

1. Κώδικας και επεξήγηση

Αρχικά, για να γίνει χρήση των μεταεργαλείων Flex και Bison έγινε μετατροπή του κώδικα της εκφώνησης από EBNF σε BNF. Ο εν λόγω κώδικας χωρίστηκε σε 2 αρχεία, ένα για τον Flex και έναν για τον Bison, που είναι τα παρακάτω:

Κώδικας αρχείου flex.l για το μεταεργαλείο Flex

```
%{  
  
#include "bison.tab.h"  
  
#include <string.h>  
#include <stdlib.h>  
  
  
#define DEFAULT 0  
#define FFOR 1  
#define FWHILE 2  
  
  
FILE *output;  
  
extern int loopcounter;  
  
extern int i;  
  
extern int flag;  
  
extern int startvalue;  
extern int addvalue;  
extern int endvalue;  
  
extern char* Index;  
extern char* printer;  
  
extern int k;  
extern int k2;  
  
void copyPrint(const char s[],int choice);  
  
%}  
  
  
letter [a-zA-Z]  
  
digit [0-9]  
  
underscore _
```

%%

```
"extern"    {fprintf(output,"extern ");  
              return EXTERN;}
```

```
"void"      {fprintf(output,"void ");  
              return VOID;}
```

```
"begin"     {fprintf(output,"begin ");  
              return BEG;}
```

```
"end"       {fprintf(output,"end ");  
              return END;}
```

```
"if"        {if(flag == DEFAULT || flag == FWHILE)  
              printf(output,"if ");  
              else  
                copyPrint(yytext,1);  
              return IF;}
```

```
"else"      {if(flag == DEFAULT || flag == FWHILE)  
              fprintf(output,"else ");  
              else  
                copyPrint(yytext,1);  
              return ELSE;}
```

```
"return"    {fprintf(output,"return ");  
              return RETURN;}
```

```
"integer"   {fprintf(output,"integer ");  
              return INT;}
```

```
"boolean"   {fprintf(output,"boolean ");  
              return BOOL;}
```

"string"	<pre> {fprintf(output,"string "); return STRING;}</pre>
"true"	<pre> {if(flag==FFOR) copyPrint(yytext,1); else fprintf(output,"true "); return TRUE;}</pre>
"false"	<pre> {if(flag==FFOR) copyPrint(yytext,1); else fprintf(output,"false "); return FALSE;}</pre>
"("	<pre> {if(flag==FFOR) copyPrint(yytext,1); else if (k!=1) fprintf(output,"("); return OPENBRACKET;}</pre>
")"	<pre> {if(flag==FFOR) copyPrint(yytext,1); else if (k!=1) fprintf(output,")"); return CLOSEBRACKET;}</pre>
"{"	<pre> {if(flag==FFOR) { if(k==1) k=0; else copyPrint(yytext,1); }</pre>

```

else if (k==1)
    k=0;
else
    fprintf(output,"{");
return OPENBRACE;}

"}"
{if(flag == FFOR)
{
    if(k2==1)
        k2=0;
    else
        copyPrint(yytext,1);
}
else if (k!=1)
    fprintf(output,"}");
return CLOSEBRACE;}

"&"
{if(flag == FFOR)
    copyPrint(yytext,1);
else
    fprintf(output,"&");
return AMPERSAND;}

"&&"
{if(flag == FFOR)
    copyPrint(yytext,1);
else
    fprintf(output,"&&");
return AND;}

"||"
{if(flag == FFOR)
    copyPrint(yytext,1);
else
    fprintf(output,"||");
return OR;}

```

"!"	<pre> {if(flag == FFOR) copyPrint(yytext,1); else fprintf(output,"!"); return NOT;}</pre>
"="	<pre> {if(flag == FFOR) copyPrint(yytext,1); else if (k!=1) fprintf(output,"="); return ASSIGN;}</pre>
"=="	<pre> {if(flag == FFOR) copyPrint(yytext,1); else fprintf(output,"=="); return EQUAL;}</pre>
"!="	<pre> {if(flag == FFOR) copyPrint(yytext,1); else fprintf(output,"!="); return NON_EQUAL;}</pre>
"<"	<pre> {if(flag == FFOR) copyPrint(yytext,1); else if (k!=1) fprintf(output,"<"); return LESS;}</pre>
">"	<pre> {if(flag == FFOR) copyPrint(yytext,1); else fprintf(output,">");</pre>

```

return GREATER;}

"<="      {if(flag == FFOR)
            copyPrint(yytext,1);
          else
            fprintf(output,"<=");
          return L_EQUAL;}

">="      {if(flag == FFOR)
            copyPrint(yytext,1);
          else
            fprintf(output,">=");
          return GR_EQUAL;}

"+"       {if(flag == FFOR)
            copyPrint(yytext,1);
          else if (k!=1)
            fprintf(output,"+");
          return ADD;}

"-"       {if(flag == FFOR)
            copyPrint(yytext,1);
          else
            fprintf(output,"-");
          return SUB;}

"*"       {if(flag == FFOR)
            copyPrint(yytext,1);
          else
            fprintf(output,"*");
          return MUL;}

"/"       {if(flag == FFOR)
            copyPrint(yytext,1);

```


	<pre> else fprintf(output,"/"); return DIV;} </pre>
"%"	<pre> {if(flag == FFOR) copyPrint(yytext,1); else fprintf(output,"%%"); return MOD;} </pre>
","	<pre> {if(flag == FFOR) copyPrint(yytext,1); else if (k!=1) fprintf(output,","); return SEMICOLON;} </pre>
","	<pre> {if(flag == FFOR) copyPrint(yytext,1); else fprintf(output,","); return COMMA;} </pre>
"for"	<pre> {flag==FFOR; k=1; return FOR;} </pre>
"while"	<pre> {if(flag == FFOR) copyPrint(yytext,1); else fprintf(output,"while"); return WHILE;} </pre>
[\n] +	<pre> {if(flag == FFOR) copyPrint(yytext,1); </pre>

```

else
    fprintf(output, "\n");
yylineno++;}

[\t]+    {if(flag == FFOR)
          copyPrint(yytext,1);
          else
          fprintf(output, "\t");}

{digit}+ {if(flag == FFOR)
          copyPrint(yytext,1);
          else if (k!=1)
          fprintf(output, "%s", yytext);
          return STATHERAAKERAIOU;}

{letter}({letter}|{digit}|{underscore})*
          {if(flag == FFOR)
          copyPrint(yytext,1);
          else if (k!=1)
          fprintf(output, "%s", yytext);
          return ID;}

({letter}|{digit}|{underscore})*
          {if(flag == FFOR)
          copyPrint(yytext,1);
          else if (k!=1)
          fprintf(output, "%s", yytext);
          return STATHERASUMBOLOSEIRA;}

%%

```

Κώδικας αρχείου bison.y για το μεταεργαλείο Bison

```
%{  
  
#include <stdio.h>  
  
#include <string.h>  
  
#include <stdlib.h>  
  
  
#define DEFAULT 0  
  
#define FFOR 1  
  
#define FWHILE 2  
  
  
int flag = DEFAULT;  
  
int i,j;  
  
char c;  
  
int k = 0 ;  
  
int k2 = 0;  
  
FILE *fp ;  
  
int loopcounter=0;  
  
int linesWithError[100];  
  
int startvalue=0;  
  
int addvalue=0;  
  
int endvalue=0;  
  
char *Index;  
  
char *printer;  
  
char *copier;  
  
char* tmp = "tmp";  
  
static char buffer[200];  
  
  
  
int errors=0;  
  
extern int lines;  
  
extern char* yytext;  
  
extern int yylineno;  
  
extern int yylval;  
  
extern FILE *yyin;  
  
extern FILE *yyout;
```

```
void copyPrint(const char s[],int choice);  
void yyerror(const char* );  
void copyFunc(char* );  
int yylex(void);  
extern FILE *output;
```

```
%}
```

```
%define parse.error verbose
```

```
%token EXTERN
```

```
%token VOID
```

```
%token BEG
```

```
%token END
```

```
%token IF
```

```
%token ELSE
```

```
%token RETURN
```

```
%token INT
```

```
%token BOOL
```

```
%token STRING
```

```
%token TRUE
```

```
%token FALSE
```

```
%token OPENBRACKET
```

```
%token CLOSEBRACKET
```

```
%token OPENBRACE
```

```
%token CLOSEBRACE
```

```
%token AMPERSAND
```

```
%token AND
```

```
%token OR
```

```
%token NOT
```

%token EQUAL

%token NON_EQUAL

%token LESS

%token GREATER

%token L_EQUAL

%token GR_EQUAL

%token ADD

%token SUB

%token MUL

%token DIV

%token MOD

%token SEMICOLON

%token COMMA

%token ID

%token STATHERASUMBOLOSEIRA

%token STATHERAAKERAIOU

%token WHILE

%token FOR

%left NOT

%left MUL DIV MOD

%left SUB ADD

%left EQUAL NON_EQUAL

%left GREATER GR_EQUAL LESS L_EQUAL

%left OR AND

%right ASSIGN

%nonassoc LOWER_THAN_ELSE

%nonassoc ELSE

%start programma

%%

programma: ekswterikesdilwseis kefalidaprogrammatos tmimaorismwn tmimaentolwn;

ekswterikesdilwseis: ekswterikoprwtotupo2;

ekswterikoprwtotupo2: %empty

| ekswterikoprwtotupo ekswterikoprwtotupo2;

ekswterikoprwtotupo: EXTERN prwtotuposunartisis;

kefalidaprogrammatos: VOID ID OPENBRACKET CLOSEBRACKET

| VOID ID OPENBRACKET error CLOSEBRACKET {yyclearin; yyerrok;};

tmimaorismwn: orismos2;

orismos2: %empty

| orismos orismos2;

orismos: orismosmetablitwn

| orismossunartisis

| prwtotuposunartisis;

orismosmetablitwn: tuposdedomenwn_metavlites SEMICOLON;

tuposdedomenwn_metavlites: INT ID

| BOOL ID

| STRING ID

| tuposdedomenwn_metavlites COMMA ID;

orismossunartisis: kefalidasunartisis tmimaorismwn tmimaentolwn;

prwtotuposunartisis: kefalidasunartisis SEMICOLON;

kefalidasunartisis: typos_synartisis_metavlites OPENBRACKET c CLOSEBRACKET

| typos_synartisis_metavlites OPENBRACKET error CLOSEBRACKET{yyclearin;
yyerrok;;

typos_synartisis_metavlites:INT ID

| BOOL ID

| VOID ID;

c: %empty

| listatupikwnparametrwn;

listatupikwnparametrwn: tupikesparametroi ntp;

ntp: %empty

| ptp ntp;

ptp: COMMA tupikesparametroi;

tupikesparametroi: tuposdedomenwn ID

| tuposdedomenwn AMPERSAND ID;

tuposdedomenwn: INT

| BOOL

| STRING;

tmimaentolwn: BEG command END;

command: %empty

| entoli command;

entoli: aplientoli SEMICOLON

| domimenientoli

| sunthetientoli

| entolifor
| entoliwhile;

sunthetientoli: OPENBRACE command CLOSEBRACE;

domimenientoli: entoliif;

aplientoli: anatesi

| klisunartisis
| entolireturn
| entolinull;

entoliif: IF OPENBRACKET genikiekfrasi CLOSEBRACKET entoli %prec LOWER_THAN_ELSE
| IF OPENBRACKET genikiekfrasi CLOSEBRACKET entoli ELSE entoli;

anatesi: ID ASSIGN genikiekfrasi

| ID error genikiekfrasi
| ID ASSIGN genikiekfrasi error ';' {yyclearin; yyerrok; };

klisunartisis: ID OPENBRACKET CLOSEBRACKET

| ID OPENBRACKET listapragmatikwnparametrwn CLOSEBRACKET;

listapragmatikwnparametrwn: pragmatikiparametros npp

| pragmatikiparametros error {yyclearin; yyerrok; };

npp: %empty

| ppp npp;

ppp: COMMA pragmatikiparametros;

pragmatikiparametros: genikiekfrasi;

entolireturn: RETURN

| RETURN genikiekfrasi;

entolinull: " ";

genikiekfrasi: genikosoros ngo;

ngo: %empty

| pgo ngo;

pgo: OR genikosoros;

genikosoros: genikosparagontas ngp;

ngp: %empty

| pgp ngp;

pgp: AND genikosparagontas;

genikosparagontas: genikosprwtparag

| NOT genikosprwtparag;

genikosprwtparag: apliekfrasi

| apliekfrasi tmimasugkrisis;

tmimasugkrisis: sugkritikostelestis apliekfrasi;

sugkritikostelestis: EQUAL

| NON_EQUAL

| LESS

| GREATER

| L_EQUAL

| GR_EQUAL;

apliekfrasi: aplosoros nao1;

nao1: %empty

| pao1 nao1;

pao1: addsub aplosoros;

addsub: ADD

| SUB;

aplosoros: aplospargontas nao2;

nao2: %empty

| pao2 nao2;

pao2: muldivmod aplospargontas;

muldivmod: MUL

| DIV

| MOD;

aplospargontas: aplospwrtoros

| addsub aplospwrtoros;

aplospwrtoros: ID

| stathera

| klisisunartisis

| OPENBRACKET genikiekfrasi CLOSEBRACKET;

stathera: STATHERAAKERAIUO

| STATHERASUMBOLOSEIRA

| TRUE

| FALSE;

entolifor: FOR OPENBRACKET ID {Index = strdup(yytext);} ASSIGN STATHERAAKERAIOU

{startvalue=atoi(yytext);} SEMICOLON ID LESS STATHERAAKERAIOU

{endvalue=atoi(yytext);}

SEMICOLON ID ASSIGN ID ADD STATHERAAKERAIOU {addvalue=atoi(yytext);}

CLOSEBRACKET

{

loopcounter =(endvalue-startvalue)/addvalue;

if(loopcounter>3)

{

k = 1 ;

flag = FWHILE;

fprintf(output,"%s = %d ;\n",Index,startvalue);

fprintf(output,"while (%s < %d) ",Index,endvalue);

fprintf(output,"{ \n %s = %s + %d;",Index,Index,addvalue);

flag=DEFAULT;

}

else

{

k2=1;

k=1;

flag = FFOR;

}

}

OPENBRACE

command {k=1;}

CLOSEBRACE {

if (flag == FFOR)

copyPrint(tmp,2);

flag=DEFAULT; };

entoliwhile: WHILE OPENBRACKET ID sugkritikostelestis STATHERAAKERAIOU CLOSEBRACKET
entoli;

%%

```

void yyerror(char const *s)
{
    fprintf(stderr, "%s\n", s);
    linesWithError[errors]=yylineno;
    errors++;
    yyclearin;
}

int main(int argc, char *argv[])
{
    fp = fopen (argv[1],"r");
    output = fopen("output.txt", "w");
    if(!fp)
    {
        printf("Can't open file \n");
        return -1;
    }
    yyin = fp;
    do {
        yyparse();
    }
    while (!feof(yyin));
    if(errors==0)
        printf("Program Parsed Successfully! \n");
    else
    {
        printf("There were %d parsing errors \n",errors);
        for(i=0;i<errors;i++)
            printf("Syntax Error on line %d \n",linesWithError[i]);
    }
    fclose(fp);
    fclose(output);
}

```

```

void copyPrint(const char s[],int choice)
{
    char *temp;
    char *tokenize[1000];
    int position=0;
    if(choice == 1)
    {
        copier = strdup(s);
        strcat(buffer,copier);
        strcat(buffer," ");
    }
    else if(choice == 2)
    {
        int met = 0,j = 0;
        temp = strdup(buffer);
        printer= strtok(temp," ");
        while(printer!=NULL)
        {
            tokenize[met] = printer;
            met++;
            printer = strtok(NULL," ");
        }
        for(i=startvalue;i<endvalue;i=i+addvalue)
        {
            for (j=0;j<met;j++)
            {
                if (!strcmp(Index,tokenize[j]) && strcmp("=",tokenize[j+1]))
                    fprintf(output,"%d ",i);
                else
                    fprintf(output,"%s ",tokenize[j]);
            }
        }
    }
}

```

Αξιοσημείωτα κομμάτια της παραπάνω υλοποίησης είναι τα εξής:

Αρχικά, για την εξάλειψη του shift/reduce conflict που μας παρουσιάστηκε στον κανόνα entoliif έγινε χρήση του %nonassoc LOWER_THAN_ELSE και %nonassoc ELSE. Σύμφωνα με το ^[1], δηλώνουμε προτεραιότητα στον τελεστή ELSE και έτσι ωθούμε τον bison να μην κάνει reduction αλλά να κάνει shift.

Όσον αφορά το πρακτικό μέρος του κώδικα, προκειμένου να δημιουργηθεί το αρχείο εξόδου, ό,τι αναγνωρίζει ο flex στο αρχείο εισόδου το εγγράφει στο νέο αρχείο με χρήση της συνάρτησης fprintf. (Μοναδική διαφοροποίηση υπάρχει στη συμπεριφορά του flex κατά την αναγνώριση δομής επανάληψης for το οποίο αναλύεται παρακάτω).

Στην συνέχεια, για να γίνει ο έλεγχος των λαθών τροποποιήθηκε η συνάρτηση yyerror() προκειμένου να επιστέφεται κατάλληλο διαγνωστικό μήνυμα.

Για την εκτίμηση του αριθμού των λαθών χρησιμοποιήθηκε το keyword error και έγινε προσπάθεια για να γίνει error recovery με την χρήση των συναρτήσεων yyerrok και yyclearin, οι οποίες συνεχίζουν το parsing της εισόδου και καθαρίζουν τα lookahead symbols αντίστοιχα. ^{[2] [3]}

Για την προσθήκη της εντολής for δημιουργήθηκε ο παρακάτω κανόνας:

```
FOR OPENBRACKET ID ASSIGN STATHERAAKERAIU SEMICOLON ID  
LESS STATHERAAKERAIU SEMICOLON ID ASSIGN ID ADD  
STATHERAAKERAIU CLOSEBRACKET OPENBRACE command  
CLOSEBRACE
```

Ενώ, για την προσθήκη της εντολής while δημιουργήθηκε ο ακόλουθος:

```
WHILE OPENBRACKET ID sugkritikostelestis STATHERAAKERAIU  
CLOSEBRACKET entoli
```

Για να υλοποιήσουμε το loop handling έγιναν τα εξής :

Αρχικά, προκειμένου να μπορούμε να υπολογίσουμε τον αριθμό των επαναλήψεων της κάθε for, προστέθηκαν actions, τα οποία αποθηκεύουν την αρχική τιμή, την τελική τιμή και το βήμα της επανάληψης. Ο τύπος που χρησιμοποιήθηκε είναι $\frac{\text{τελική τιμή} - \text{αρχική τιμή}}{\text{βήμα}}$. Για να μπορεί να υπολογιστεί

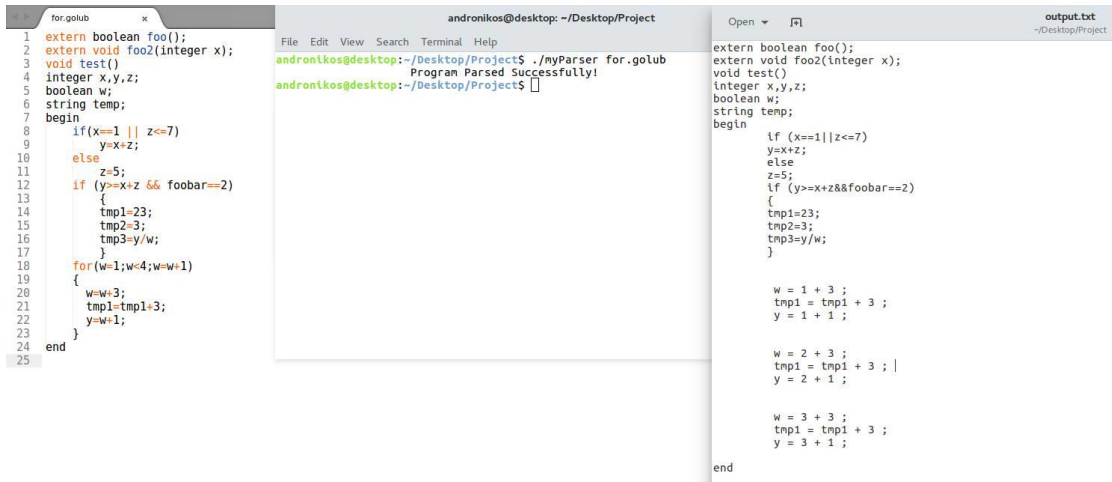
ντετερμινιστικά το αποτέλεσμα, η μόνη μορφή `for` που αναγνωρίζεται είναι αυτή που η αρχική τιμή είναι μικρότερη της τελικής και το βήμα αυξάνεται αθροιστικά. Παράλληλα, προκειμένου να μπορέσουμε να επιτύχουμε την σωστή αντικατάσταση του μετρητή της επανάληψης, τον αποθηκεύσαμε στην μεταβλητή `Index`.

Στην περίπτωση που ο αριθμός των επαναλήψεων είναι μεγαλύτερος του 3 τυπώνεται στο πρόγραμμα εξόδου η αρχικοποίηση του μετρητή επανάληψης, η επικεφαλίδα της `while` με την συνθήκη αλλά και η αύξηση του μετρητή και στην συνέχεια ο κώδικας του σώματος της `for`. Προκειμένου να μην εκτυπωθεί η αγκύλη της `for` εισόδου αλλά και η επικεφαλίδα της κάνουμε την μεταβλητή `k = 1`, γεγονός που απενεργοποιεί όλες τις εγγραφές στο αρχείο εισόδου και το ενεργοποιούμε ξανά μετά την λεκτική εύρεση του συμβόλου `{`. Με αυτό τον τρόπο δεν εμφανίζεται στην έξοδο το `brace` που ανοίγει την εντολή `for`.

Στην περίπτωση που ο αριθμός των επαναλήψεων είναι μικρότερος ή ίσος του 3 εκτελούμε `loop unrolling`. Η λειτουργία που εκτελούμε είναι παρόμοια με της προηγούμενης περίπτωσης, με την διαφορά ότι υπάρχει μια δεύτερη μεταβλητή η `k2`, η οποία όπως και η μεταβλητή `k`, μόλις πάρει την τιμή 1 απενεργοποιεί την εμφάνιση του συμβόλου `}`. Με αυτό τον τρόπο δεν εμφανίζεται στην έξοδο το `brace` που κλείνει την εντολή `for`. Επιπρόσθετα, μιας και θέλουμε να τυπώσουμε μία, δύο ή τρεις φορές το σώμα της `for` με τις όποιες αλλαγές θέλουμε να γίνουν, μέσω του `flex` στέλνουμε στην συνάρτηση `copyPrint` όλα τα `tokens` που διαβάζονται, και αφού τους προσθέσουμε ένα κενό στο τέλος, τα αποθηκεύουμε σε ένα πίνακα με `strings`. Όταν τελειώσει το σώμα της `for` ξανακαλούμε την συνάρτηση αυτή και αποθηκεύουμε σε ένα δισδιάστατο πίνακα όλα τα `tokens` (το κάθε ένα σε ξεχωριστή γραμμή). Ύστερα με μια επαναληπτική δομή, προελαύνουμε τον παραπάνω πίνακα και συγκρίνουμε το κάθε `string` με τον μετρητή του αρχείου εισόδου. Αν είναι το ίδιο και δεν ακολουθείται από `=` τότε το αντικαθιστούμε με τον μετρητή της επαναληπτικής μας δομής, αλλιώς, όπως και στην γενική περίπτωση το τυπώνουμε όπως το διαβάσαμε στην είσοδο.

2. Αποτελέσματα

Τα αποτελέσματα που προκύπτουν με βάση την παραπάνω γραμματική είναι:



```
1 extern boolean foo();
2 extern void foo2(integer x);
3 void test()
4 integer x,y,z;
5 boolean w;
6 string temp;
7 begin
8   if(x==1 || z<=7)
9     y=x+z;
10  else
11    z=5;
12  if (y>=x+z && foobar==2)
13  {
14    tmp1=23;
15    tmp2=3;
16    tmp3=y/w;
17  }
18  for(w=1;w<4;w=w+1)
19  {
20    w=w+3;
21    tmp1=tmp1+3;
22    y=w+1;
23  }
24 end
25
```

```
andronikos@desktop: ~/Desktop/Project
File Edit View Search Terminal Help
andronikos@desktop:~/Desktop/Project$ ./myParser for.golub
Program Parsed Successfully!
andronikos@desktop:~/Desktop/Project$
```

```
extern boolean foo();
extern void foo2(integer x);
void test()
integer x,y,z;
boolean w;
string temp;
begin
  if (x==1||z<=7)
    y=x+z;
  else
    z=5;
  if (y>=x+z&&foobar==2)
  {
    tmp1=23;
    tmp2=3;
    tmp3=y/w;
  }

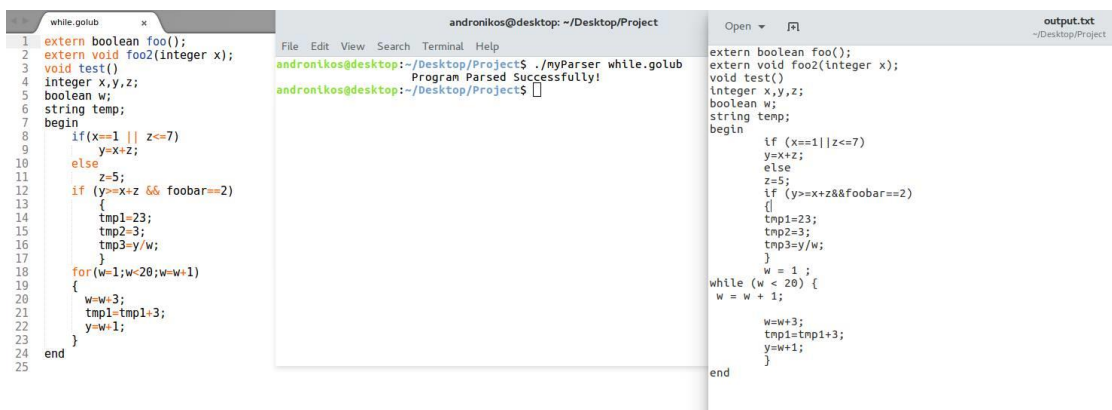
  w = 1 + 3 ;
  tmp1 = tmp1 + 3 ;
  y = 1 + 1 ;

  w = 2 + 3 ;
  tmp1 = tmp1 + 3 ; |
  y = 2 + 1 ;

  w = 3 + 3 ;
  tmp1 = tmp1 + 3 ;
  y = 3 + 1 ;

end
```

Εικόνα 1^η: Είσοδος χωρίς συντακτικά λάθη με loop-unrolling



```
1 extern boolean foo();
2 extern void foo2(integer x);
3 void test()
4 integer x,y,z;
5 boolean w;
6 string temp;
7 begin
8   if(x==1 || z<=7)
9     y=x+z;
10  else
11    z=5;
12  if (y>=x+z && foobar==2)
13  {
14    tmp1=23;
15    tmp2=3;
16    tmp3=y/w;
17  }
18  for(w=1;w<20;w=w+1)
19  {
20    w=w+3;
21    tmp1=tmp1+3;
22    y=w+1;
23  }
24 end
25
```

```
andronikos@desktop: ~/Desktop/Project
File Edit View Search Terminal Help
andronikos@desktop:~/Desktop/Project$ ./myParser while.golub
Program Parsed Successfully!
andronikos@desktop:~/Desktop/Project$
```

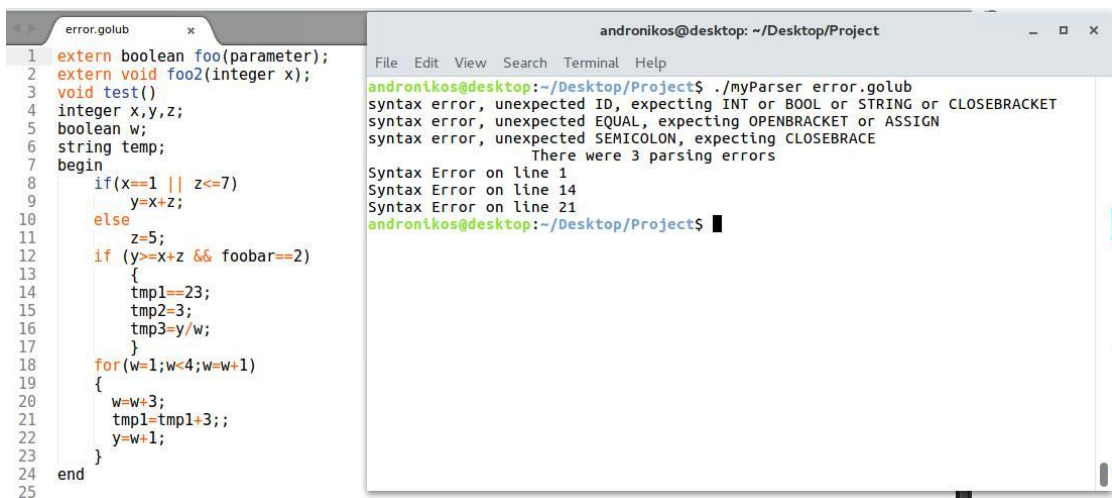
```
extern boolean foo();
extern void foo2(integer x);
void test()
integer x,y,z;
boolean w;
string temp;
begin
  if (x==1||z<=7)
    y=x+z;
  else
    z=5;
  if (y>=x+z&&foobar==2)
  {
    tmp1=23;
    tmp2=3;
    tmp3=y/w;
  }

  w = 1 ;
  while (w < 20) {
    w = w + 1;

    w=w+3;
    tmp1=tmp1+3;
    y=w+1;
  }

end
```

Εικόνα 2^η: Είσοδος χωρίς συντακτικά λάθη και αντικατάσταση της for με while



```
1 extern boolean foo(parameter);
2 extern void foo2(integer x);
3 void test()
4 integer x,y,z;
5 boolean w;
6 string temp;
7 begin
8   if(x==1 || z<=7)
9     y=x+z;
10  else
11    z=5;
12  if (y>=x+z && foobar==2)
13  {
14    tmp1==23;
15    tmp2=3;
16    tmp3=y/w;
17  }
18  for(w=1;w<4;w=w+1)
19  {
20    w=w+3;
21    tmp1=tmp1+3;;
22    y=w+1;
23  }
24 end
25
```

```
andronikos@desktop: ~/Desktop/Project
File Edit View Search Terminal Help
andronikos@desktop:~/Desktop/Project$ ./myParser error.golub
syntax error, unexpected ID, expecting INT or BOOL or STRING or CLOSEBRACKET
syntax error, unexpected EQUAL, expecting OPENBRACKET or ASSIGN
syntax error, unexpected SEMICOLON, expecting CLOSEBRACE
There were 3 parsing errors
Syntax Error on line 1
Syntax Error on line 14
Syntax Error on line 21
andronikos@desktop:~/Desktop/Project$
```

Εικόνα 3^η: Είσοδος με συντακτικά λάθη στις γραμμές 1,14 και 21

Βιβλιογραφία

[1] John R. Levine, *flex & bison*, 1st edition, O'Reilly Media, Inc, 2009, pg. 188

[2] **Free Software Foundation, 6. Error Recovery**

https://www.gnu.org/software/bison/manual/html_node/Error-Recovery.html

[3] **Free Software Foundation, 2.3 Simple Error Recovery**

https://www.gnu.org/software/bison/manual/html_node/Simple-Error-Recovery.html