Αναφορά Εργασίας Αρχών Γλωσσών Προγραμματισμού & Μεταφραστών

Βένος Αναστάσιος ΑΜ:1067536 3ο έτος email:up1067536@upnet.gr

Ζερβού Πουλχερία ΑΜ:1067511 3ο έτος email:up1067511@upnet.gr

Πανδής Αναστάσιος ΑΜ: 1056287 4ο έτος email:up1056287@upnet.gr

Πορτοκάλογλου Ανδρονίκη ΑΜ:1067539 3ο έτος email:up1067539@upnet.gr

1)Περιγραφή Γραμματικής BNF

Αρχικά, ορίζουμε μέσω του code την μορφή που θα έχει ο κώδικας μας. Βάζουμε πρώτα την δεσμευμένη λέξη PROGRAM, έπειτα το Args μέσω του οποίου ορίζουμε το όνομα του προγράμματος, το function_definition το οποίο αποτελεί την γραμματική για τον ορισμό των συναρτήσεων και το main_definition στο οποίο ορίζεται η μορφή της main.

Function_definition:

Ξεκινάει με το FUNCTION, έπειτα μέσω του name ορίζεται το όνομα της συνάρτησης, έπειτα στο parameters ορίζονται οι παράμετροι της συνάρτησης. Έπειτα έχουμε αλλαγή γραμμής και καθορισμό των μεταβλητών. Στην συνέχεια, ακολουθούν οι εντολές. Έχουμε τις δεσμευμένες λέξεις RETURN και END_FUNCTION.

Main_definition:

Ξεκινάει με την δεσμευμένη λέξη STARTMAIN και συνεχίζει με ορισμό μεταβλητών και τις εμπεριεχόμενες εντολές. Τελειώνει με την δεσμευμένη λέξη ENDMAIN.

Commands:

Στις εντολές ανήκουν το assign, loop, check, print, break, comments.

Assign:

Η γραμματική του variable χρησιμοποιώντας έναν assign operator που είναι το ίσον (=) σε συνδυασμό έπειτα με μια complex expression. Η γραμματική του complex expression αποτελείται από εκφράσεις προσθέσεων ή αφαίρεσης ή πολλαπλασιασμών ή διαιρέσεων ή εκθετικής ύψωσης, με ή χωρίς παρενθέσεις.

Break:

Αποτελείται από την δεσμευμένη λέξη BREAK.

Print:

Αποτελείται από την δεσμευμένη λέξη print, παρενθέσεις μέσα στις οποίες υπάρχουν σταθερές, ονόματα, args, τα οποία είναι τα strings που εκτυπώνονται από την print, δίπλα στα οποία καλούνται οι μεταβλητές.

Comments:

Η γραμματική των σχολίων της ίδιας γραμμής αποτελείται από το σύμβολο % και μέσω της γραμματικής value επιλέγονται τα σχόλια που θέλει να ορίσει ο χρήστης.

Loop:

Αποτελείται από 2 υπό-γραμματικές, while και for.

While:

Ξεκινάει με την δεσμευμένη λέξη while, και στις παρενθέσεις ορίζεται η συνθήκη. Η γραμματική της συνθήκης αποτελείται από 2 εκφράσεις ανάμεσα στις οποίες υπάρχει ένας λογικός τελεστής. Μετά την συνθήκη υπάρχει αλλαγή γραμμής, καλούνται οι εντολές που θα βρίσκονται μέσα στην for αλλάζοντας γραμμή ανά εντολή. Τελειώνει με την δεσμευμένη λέξη ENDWHILE.

For:

Ξεκινάει με την δεσμευμένη λέξη for. Η δεσμευμένη λέξη counter η οποία ισούται με έναν αριθμό από την γραμματική constant. Έχουμε τις δεσμευμένες λέξεις to, step στις οποίες ορίζεται η συχνότητα με την οποία θα εκτελεστεί η for μας. Στην συνέχεια έχουμε αλλαγή γραμμής. Έπειτα καλούνται τα commands και τελειώνει με την δεσμευμένη λέξη ENDFOR.

Check:

Η γραμματική check αποτελείται από 2 υπογραμματικές, check1 και check2. Το πρώτο περιέχει την if, ενώ το δεύτερο την switch.

If:

Ξεκινάει με την δεσμευμένη λέξη if. Μέσα σε παρενθέσεις υπάρχει η συνθήκη με την οποία θα εκτελείται. Έπειτα με την δεσμευμένη λέξη THEN συνεχίζει στα commands. Μέσα σε παρενθέσεις και με την χρήση του Kleene star ορίζουμε τα else ενδεχόμενα της if. Έχουμε την δεσμευμένη λέξη else if, και δίπλα μέσα σε παρενθέσεις την συνθήκη, ενώ έπειτα ακολουθούν ξανά

commands. Στην συνέχεια ακολουθεί η δεσμευμένη λέξη else με τα αντίστοιχα commands που εμπεριέχει. Η if κλείνει με την δεσμευμένη λέξη END IF.

Switch:

Ξεκινάει με την λέξη SWITCH και μέσα σε παρενθέσεις βρίσκεται η έκφραση (complex expression). Έπειτα ακολουθεί η λέξη CASE (complex expression), το σύμβολο της άνω κάτω τελείας και τις καλούμενες commands. Έπειτα ακολουθεί η δεσμευμένη λέξη DEFAULT με άνω κάτω τελεία ακολουθούμενη από εντολές. Η γραμματική τελειώνει με την δεσμευμένη λέξη ENDSWITCH.

Struct:

Ξεκινάει με την δεσμευμένη λέξη STRUCT και ακολουθεί το όνομα τύπου του struct. Έχουμε αλλαγή γραμμής και ορισμό των μεταβλητών όπως περιγράφεται παρακάτω. Η γραμματική κλείνει με την δεσμευμένη λέξη ENDSTRUCT. Εναλλακτικά, η γραμματική αυτή μπορεί να χρησιμοποιηθεί με την λέξη TYPEDEF πριν την λέξη STRUCT και το όνομα τύπου πριν την λέξη ENDSTRUCT.

Μεταβλητές:

<var-declare>::=VARS <variables>*<semicolon> Ορισμός μεταβλητών

<variables>::=<characters> Μεταβλητές

|<integers>

<characters>::=CHAR<name>+ Χαρακτήρες

<integers>::=INTEGER<name>+ Ακέραιοι

<name>::= <Args><constant>*<coma>*

|<ARRAY>*<coma>*

<ARRAY>::=<Args><constant>* <'['constant']'> Πίνακες

Μικρές γραμματικές που βοήθησαν στην εκτέλεση των παραπάνω:

<Args>::=[a-zA-Z] Ορισμός strings

<coma>::=","

<constant>::=[0-9]

<semicolon>::=";"

<value>::= <name>

| <constant>

| <Args>

<logexp>::= >|<|==|!=|AND|OR Λογικοί τελεστές

<sinthiki>::= <value> <logexo> <value> Ορισμός λογικής έκφρασης

<mul-operators>::= "^"|"*"|"/" Αριθμητοί τελεστές

<ad-operators>::="+"|"-" Αριθμητοί τελεστές

<assign-operator>::="="

<parameters>::=<Args><constant>*

Λεξικός αναλυτής (flex):

Στο αρχείο flex ορίζουμε όλες τις δεσμευμένες λέξεις της ψευδό-γλώσσα μας. Αρχικά, μέσα σε αυτάκια βάζουμε όλα τα σύμβολα που χρησιμοποιούμε και επιστρέφουμε το token που τα συνδέει με το bison. Στην ίδια λογική, ορίζουμε όλες τις δεσμευμένες λέξεις και επιστρέφουμε όλα τα token τους. Κάποια σημεία που αξίζει να εστιάσουμε είναι ο ορισμός αλλαγής γραμμής στον οποίο χρησιμοποιείται το yyline της flex όπου εκτυπώνεται και ο αριθμός της γραμμής ώστε να εάν υπάρξει συντακτικό σφάλμα (syntax error) να ξέρουμε σε ποια γραμμή είναι. Ακόμη, χρησιμοποιούμε το atoi για να αναπαραστήσουμε τους αριθμούς με περισσότερο από ένα ψηφιά και αντίστοιχα το strdup για αναπαράσταση strings περισσοτέρων από ένα χαρακτήρων.

Συντακτικός αναλυτής (bison):

Είσοδοι flex&bison

1η Είσοδος:

PROGRAM code

STARTMAIN

VARS

INTEGER num,var;

SWITCH(var+1)

CASE(1)

PRINT("caseone"[,num]);

CASE(2)

PRINT("casetwo"[,num]);

DEFAULT

PRINT("casethree"[,num]);

ENDSWITCH

ENDMAIN

Στην παραπάνω είσοδο χρησιμοποιούνται οι γραμματικές των:

Code:

```
code: PRGM name NEWLINE struct function mainDefiniiton;
```

Main:

```
//main syntax
mainDefiniiton: SMN NEWLINE declarationstar NEWLINE prog_commands EMN
              ;
```

Declarationstar:

```
declarationstar: |declaration
               ;
```

Declaration:

```
declaration: VAR NEWLINE variables  SEMI;  //variable declaration syntax
```

Variables:

```
//variables syntax
variables:  variables variables
          |integers
          |characters
          ;
```

Integer & characters:

```
characters: ABC  name   //char syntax
          |characters array
          ;
integers: INT  name     //integer syntax
          |integers array
          ;
```

Name:

```
name: |name COMA str con     //name syntax (var,var1 ...)
    |name COMA str
    |str con
    |str
    ;
```

Array:

```
array: LAGY  con  RAGY ; //array syntax
```

Prog_commands:

```
//in commands commands syntax
prog_commands:  |prog_commands expr NEWLINE
              |prog_commands commands
              |expr NEWLINE
              |commands
              ;
```

Commands:

```
commands: commands commands
        |cs
        |assign
        |brk
        |comments
        |while0
        |for0
        |check1
        |check2
        |print
        |callfucntion
        ;
```

Switch(check2):

```
check2:SWTCH LPAR expr RPAR NEWLINE cs default ESWTCH NEWLINE ;

cs:     CS LPAR expr RPAR NEWLINE prog_commands
        ;

default: |DFLT NEWLINE prog_commands
         ;
```

Που χρησιμοποιεί και τις γραμματικές case (cs) και default.

Expr:

```
//expresion syntax
expr :  expr ADD  expr
       | expr MIN  expr
       | expr DIV  expr        {if($3==0)yyerror("\ncan't divide with zero.");}
       | expr MUL  expr
       | LPAR expr RPAR       {$$=$2;}
       | con
       |name
       ;
```

Print:

```
//print syntax
print:PRT LPAR QM value QM RPAR SEMI NEWLINE
        |PRT LPAR QM value QM LAGY COMA name RAGY RPAR SEMI NEWLINE
        ;
```

Value:

```
value:  con      //returned value can be a constant, a string or a name
        |name
        ;
```

Εφαρμογή 1ης εισόδου:

```
$ ./parser input2.txt
line:1
line:2
line:3
line:4
line:5
line:6
line:7
line:8
line:9
line:10
line:11
line:12
Enter the filename to open
input2.txt
PROGRAM code
STARTMAIN
VARS
INTEGER num,var;
SWITCH(var+1)
CASE(1)
PRINT("caseone"[,num]);
CASE(2)
PRINT("casetwo"[,num]);
DEFAULT
PRINT("casethree"[,num]);
ENDSWITCH
ENDMAIN
```

2η Είσοδος:

```
PROGRAM code
FUNCTION hlikia(age)
VARS
INTEGER age18;
WHILE(age!=0)
IF(age>age18)THEN
PRINT("adult");
ELSE
PRINT("underage");
ENDIF
BREAK;
ENDWHILE
RETURN 0
ENDFUNCTION
STARTMAIN
VARS
INTEGER choseage;
choseage=5;
hlikia(choseage);
ENDMAIN
```

Στην παραπάνω είσοδο εκτός από τις γραμματικές που προαναφέρθηκαν χρησιμοποιούνται επίσης οι γραμματικές των:

Function:

```
//function syntax
function: |DEF name LPAR parameters RPAR NEWLINE declarationstar NEWLINE prog_commands RTRN value NEWLINE EDEF NEWLINE;
```

Parameters:

```
parameters:     |parameters COMA str con
                |parameters COMA str
                |str con
                |str
                ;
```

While:

```
//while syntax
while0: WHL LPAR condition RPAR NEWLINE prog_commands EWHL NEWLINE;
```

If (check1):

```
check1:LIF LPAR condition RPAR THN NEWLINE prog_commands EIF NEWLINE
       |LIF LPAR condition RPAR THN NEWLINE prog_commands elseif EIF NEWLINE
       |LIF LPAR condition RPAR THN NEWLINE prog_commands elseif lelse EIF NEWLINE
       |LIF LPAR condition RPAR THN NEWLINE prog_commands lelse EIF NEWLINE
       ;

elseif: LELSELIF LPAR condition RPAR NEWLINE prog_commands;

lelse: LELSE NEWLINE prog_commands;
```

Condition:

```
//condition syntax
condition: condition logexp condition
        | value logexp  value
        |value
        ;
//logical operators
logexp: LLE
        |RLE
        |EQUL
        |NOTEQ
        |AND
        |OR
        ;
```

Break:

```
//break syntax
brk: BRK SEMI NEWLINE
```

Assign:

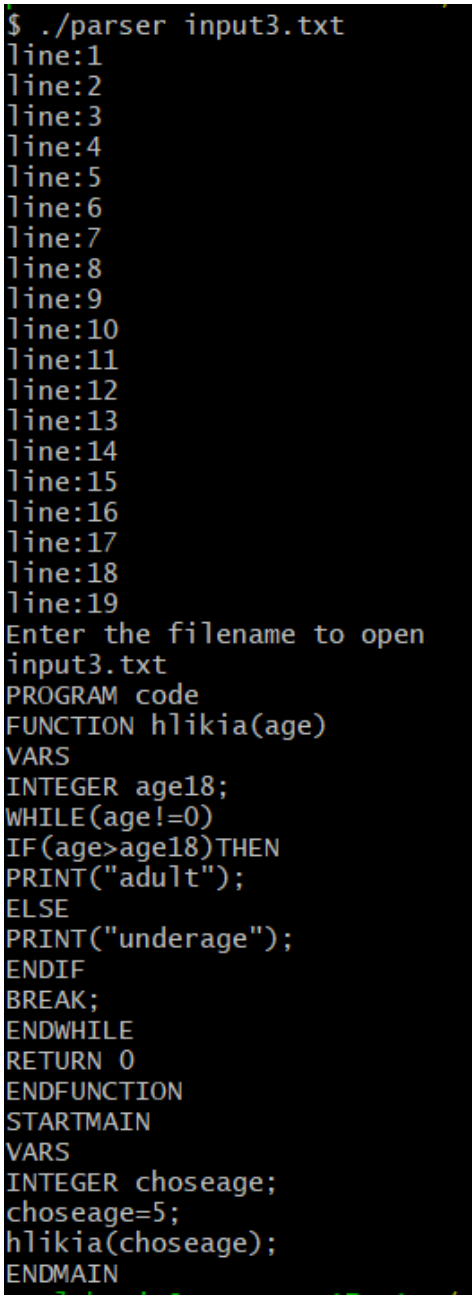
```
assign: name ASSGN expr SEMI NEWLINE
        |name array ASSGN expr SEMI NEWLINE//assign syntax
        ;
```

Callfunction:

```
callfucntion:name LPAR parameters RPAR SEMI NEWLINE;
```

Η σύνταξη callfunction δεν αναγράφεται στην εκφώνηση της εργασία και αποτελεί την σύνταξη που χρησιμεύει στο να καλέσουμε την ορισμένη συνάρτηση.

Εφαρμογή 2ης εισόδου:



Λανθασμένη Είσοδος:

PROGRAM code

STARTMAIN

INTEGER num,var;

SWITCH(var+1)

CASE(1)

PRINT("caseone"[,num]);

CASE(2)

PRINT("casetwo"[,num]);

DEFAULT

PRINT("casethree"[,num]);

ENDSWITCH

ENDMAIN

Παρατηρούμε ότι στην τρίτη γραμμή υπάρχει συντακτικό σφάλμα γιατί πριν το INTEGER num,var; θα έπρεπε να υπάρχει ορισμός μεταβλητών δηλαδή: VARS

```
$ ./parser wronginput.txt
line:1
line:2
syntax error
Enter the filename to open
```

2) Δήλωση τύπου δεδομένων χρήστη / Δήλωση δομής (struct):

Ξεκινάει με την δεσμευμένη λέξη STRUCT και ακολουθεί το όνομα τύπου του struct. Έχουμε αλλαγή γραμμής και ορισμό των μεταβλητών όπως περιγράφεται παραπάνω. Η γραμματική κλείνει με την δεσμευμένη λέξη ENDSTRUCT. Εναλλακτικά, η γραμματική αυτή μπορεί να χρησιμοποιηθεί με την λέξη TYPEDEF πριν την λέξη STRUCT και το όνομα τύπου πριν την λέξη ENDSTRUCT. Στον λεκτικό αναλυτή flex προσθέσαμε τις δεσμευμένες λέξεις “STRUCT, ENDSTRUCT, TYPEDEF", τις οποίες τις επιστρέφει ως STR, ENDSTR, TPDEF αντίστοιχα. Στον συντακτικό αναλυτή bison τα συνδέσαμε με τα tokens STR, ENDSTR, TPDEF.

```
//struct syntax
struct:     |struct STR name NEWLINE declarationstar NEWLINE ENDSTR NEWLINE
            |struct TPDEF STR name NEWLINE declarationstar name NEWLINE ENDSTR NEWLINE
            | STR name NEWLINE declarationstar NEWLINE ENDSTR NEWLINE
            | TPDEF STR name NEWLINE declarationstar name NEWLINE ENDSTR NEWLINE
            ;
```

3) Έλεγχος σωστής δήλωσης των μεταβλητών που χρησιμοποιούνται οπουδήποτε στο πρόγραμμα: Δεν υλοποιήθηκε.

4) Σχόλια πολλαπλών γραμμών:

Τα σχόλια πολλαπλών γραμμών βρίσκονται στο flex.l. Αρχίζουν με "/*" και τελειώνουν με "*/". Με το "<C_COMMENT>\n" δίνεται η δυνατότητα να εισάγουμε πολλαπλές γραμμές σχολίων.

```
%x C_COMMENT

%%

"/*"              { BEGIN(C_COMMENT); }
<C_COMMENT>"*/"   { BEGIN(INITIAL); }
<C_COMMENT>\n     { }
<C_COMMENT>.      { }
```

Η Τρίτη είσοδος χρησιμοποιεί τις γραμματικές των 2 και 4 ερωτημάτων σε συνδυασμό με τις γραμματικές που υλοποιήθηκαν στο 1 ερώτημα.

3η Είσοδος:

PROGRAM code

STRUCT str1

VARS

```
INTEGER array1[2] /* comment */ INTEGER c1,c2 CHAR c3,c4;
ENDSTRUCT
FUNCTION func(a,b,par)
VARS
INTEGER v1,v2 CHAR bathmos,v4 INTEGER array[1];
IF(y>5)THEN
bathmos=1;
ELSE
batmos=0;
ENDIF
RETURN 0
ENDFUNCTION
STARTMAIN
VARS
INTEGER v1,v2 CHAR v3,v4 INTEGER array[1];
FOR COUNTER = 1 TO 5 STEP 1
PRINT("BATHMOLOGIES"[,array]);
ENDFOR
WHILE(v1!=0)
func(v1,v2,v3);
v4=(v1+v2+v3)/2;
ENDWHILE
ENDMAIN
```

Στην παραπάνω είσοδο επιπλέον χρησιμοποιείτε η εντολή for ,η οποία δεν ξαναχρησιμοποιείτε στις προηγούμενες εισόδους με σύνταξη:

```
for0:FR CNTR ASSGN con  LTO  con STP expr NEWLINE prog_commands EFR NEWLINE;
```

Εφαρμογή 3ης εισόδου:

```
$ ./parser input.txt
line:1
line:2
line:3
line:4
line:5
line:6
line:7
line:8
line:9
line:10
line:11
line:12
line:13
line:14
line:15
line:16
line:17
line:18
line:19
line:20
line:21
line:22
line:23
line:24
line:25
Enter the filename to open
input.txt
PROGRAM code
STRUCT str1
VARS
INTEGER array1[2] /* comment */ INTEGER c1,c2 CHAR c3,c4;
ENDSTRUCT
FUNCTION func(a,b,par)
VARS
INTEGER v1,v2 CHAR bathmos,v4 INTEGER array[1];
IF(y>5)THEN
bathmos=1;
ELSE
batmos=0;
ENDIF
RETURN 0
ENDFUNCTION
STARTMAIN
VARS
INTEGER v1,v2 CHAR v3,v4 INTEGER array[1];
FOR COUNTER = 1 TO 5 STEP 1
PRINT("BATHMOLOGIES"[,array]);
ENDFOR
WHILE(v1!=0)
func(v1,v2,v3);
v4=(v1+v2+v3)/2;
ENDWHILE
ENDMAIN
```