ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Ηλεκτρονικών Υπολογιστών

ΘΕΩΡΙΑ ΥΠΟΛΟΓΙΣΜΟΥ - ΠΛΗ402
Εαρινό Εξάμηνο Ακαδημαϊκού Έτους 2016-2017

Εργασία Προγραμματισμού
Λεκτική και Συντακτική Ανάλυση
της Γλώσσας Προγραμματισμού FC

Νικόλαος Κυπαρισσάς, 2012030112

Η Λεκτική Ανάλυση (Εργαλείο Flex)

Λόγω του ότι η γλώσσα FC μοιάζει σε αρκετά σημεία με τη γλώσσα C, η λεκτική ανάλυση παρήγαγε τα περισσότερα σύμβολα αυτούσια από την είσοδο. Η χρήση κανονικών εκφράσεων περιορίστηκε στην αναγνώριση πιο σύνθετων λεκτικών δομών:

identifier: [a-zA-Z_][0-9a-zA-Z_]*
Η κανονική έκφραση επιτρέπει τη δημιουργία συμβολοσειρών όπως οι: *x*, *x0*, *xX0*, *Z_01a*. Αποκλείει την αποδοχή συμβολοσειρών που ξεκινούν από αριθμό ή σύμβολο διαφορετικό των γραμμάτων του αλφαβήτου.

positive integer: [0] | [1-9][0-9]*
Η κανονική έκφραση επιτρέπει τη δημιουργία συμβολοσειρών όπως οι: *0*, *3*, *42*, *23000001*. Αποκλείει την αποδοχή συμβολοσειρών που ξεκινούν με περιττά μηδενικά.

positive real number: {**positive integer**}"."[0-9]*([eE][+-]?{**positive integer**})?
Η κανονική έκφραση επιτρέπει τη δημιουργία συμβολοσειρών όπως οι: *42.0*, *4.2e1*, *0.420E+2*, *1234.12345678e-123*.

escape sequence: \\n | \\t | \\r | \\\\ | \\\' | \\\"

constant character: \'({**escape sequence**} | [^\\^\'^\"]) \'

string: \" ({**escape sequence**} | [^\n])* \"

line comment: "//" [^\n]*

block comment: "/*" (.*[\n]*)* "*/"

Η Γραμματική (Εργαλείο Bison)

Ακολουθούν οι κανόνες της γραμματικής μας:

$accept → *input* **$end**

input → *global_declaration*

global_declaration → *data_type* **"identifier"** **'('** *arguments_declaration* **')'** *block* *functions_declaration*
| *data_type* **"identifier"** *array_dimensions* **';'** *global_declaration*
| *data_type* **"identifier"** *array_dimensions* **"operator: assignment"** *expression* **';'** *global_declaration*
| *data_type* **"identifier"** *array_dimensions* **','** *variables_list* **';'** *global_declaration*
| *data_type* **"identifier"** *array_dimensions* **"operator: assignment"** *expression* **','** *variables_list* **';'** *global_declaration*

Ο παραπάνω κανόνας διασφαλίζει ότι το πρόγραμμα ξεκινάει είτε με δήλωση καθολικών μεταβλητών που ακολουθείται από δήλωση τουλάχιστον μίας συνάρτησης, είτε κατευθείαν με δήλωση συνάρτησης. Μετά τη δήλωση της πρώτης συνάρτησης, δεν υπάρχει τρόπος δήλωσης καθολικών μεταβλητών, αφού κάτι τέτοιο δεν επιτρέπεται από τους κανόνες της γλώσσας FC.

variables_declaration → *data_type variables_list* **';'**

variables_list → **"identifier"** *array_dimensions*
| **"identifier"** *array_dimensions* **"operator: assignment"** *expression*
| *variables_list* **','** *variables_list*

functions_declaration → %empty
| *functions_declaration data_type* **"identifier"** **'('** *arguments_declaration* **')'** *block*

data_type → *static* **"keyword: integer"**
| *static* **"keyword: boolean"**
| *static* **"keyword: character"**
| *static* **"keyword: real"**
| *static* **"keyword: void"**
| *static* **"keyword: string"**

static → %empty
| **"keyword: static"**

arguments → *expression*
| *arguments* **','** *arguments*

arguments_declaration → %empty
| *data_type* **"identifier"** *array_dimensions*
| *arguments_declaration* **','** *data_type* **"identifier"** *array_dimensions*

array_dimensions → %empty
| *array_dimensions* **'['** *arithmetic_expression* **']'**

block → **"keyword: begin"** *body* **"keyword: end"**
| *statement* **';'**
| *if_statement*
| *while_loop*
| *for_loop*
| *do_while_loop*

body → %empty
| *statement* **';'** *body*
| *variables_declaration body*
| *if_statement body*
| *while_loop body*
| *for_loop body*
| *do_while_loop body*

statement → %empty
| **"identifier"** *array_dimensions* **"operator: assignment"** *expression*
| **"identifier"** **'('** *arguments* **')'**
| **"identifier"** **'('** **')'**
| **"keyword: continue"**
| **"keyword: break"**
| **"keyword: return"**
| **"keyword: return"** *expression*

if_statement → **"keyword: if"** **'('** *expression* **')'** *block*
| **"keyword: if"** **'('** *expression* **')'** *block* **"keyword: else"** *block*

while_loop → **"keyword: while"** **'('** *expression* **')'** *block*

for_loop → **"keyword: for"** **'('** *statement* **';'** *expression* **';'** *statement* **')'** *block*
| **"keyword: for"** **'('** *statement* **';'** **';'** *statement* **')'** *block*

do_while_loop → **"keyword: do"** *block* **"keyword: while"** **'('** *expression* **')'** **';'**

expression → **"identifier"** *array_dimensions*
| **'!' "identifier"** *array_dimensions*
| **"keyword: not" "identifier"** *array_dimensions*
| **"identifier" '('** *arguments* **')'**
| **"identifier" '(' ')'**
| **"integer"**
| **"real number"**
| **"string"**
| **"constant character"**
| **"keyword: true"**
| **"keyword: false"**
| **'-'** *expression*
| **'('** *expression* **')'**
| *expression* **'+'** *expression*
| *expression* **'-'** *expression*
| *expression* **'*'** *expression*
| *expression* **"keyword: mod"** *expression*
| *expression* **'/'** *expression*
| *expression* **'<'** *expression*
| *expression* **'>'** *expression*
| *expression* **'='** *expression*
| *expression* **"comparator: not equal"** *expression*
| *expression* **"comparator: less or equal"** *expression*
| *expression* **"comparator: greater or equal"** *expression*
| *expression* **"keyword: and"** *expression*
| *expression* **"keyword: or"** *expression*
| *expression* **"operator: AND"** *expression*
| *expression* **"operator: OR"** *expression*

arithmetic_expression → **"identifier"** *array_dimensions*
| **"identifier" '('** *arguments* **')'**
| **"identifier" '(' ')'**
| **"integer"**
| **"real number"**
| **'-'** *arithmetic_expression*
| **'('** *arithmetic_expression* **')'**
| *arithmetic_expression* **'+'** *arithmetic_expression*
| *arithmetic_expression* **'-'** *arithmetic_expression*
| *arithmetic_expression* **'*'** *arithmetic_expression*
| *arithmetic_expression* **'/'** *arithmetic_expression*
| *arithmetic_expression* **"keyword: mod"** *arithmetic_expression*

Η γραμματική μας δεν οδηγεί το bison σε καμία σύγκρουση χάρη στις προτεραιότητες που θέσαμε:

%right "*simple_if*" **"keyword: else"** : αυτή η εντολή προτεραιότητας επιλύει την σύγκρουση του "dangling else". Ο ορισμός της δεν ήταν απαραίτητος, αφού το εργαλείο Bison επιλύει σωστά την σύγκρουση επιλέγοντας να μετακινήσει το "else" στη στοίβα αντί να κάνει αναγωγή. Παρ' όλα αυτά, επιλέξαμε την απαλοιφή της σύγκρουσης. Η προτεραιότητα επιβάλεται στον αντίστοιχο κανόνα της γραμματικής όπως φαίνεται παρακάτω:

```
if_statement:
     KW_IF '(' expression ')' block %prec "simple_if"
|    KW_IF '(' expression ')' block KW_ELSE block ;
```

Οι παρακάτω κανόνες προτεραιότητας διασφαλίζουν την αποφυγή συγκρούσεων που δημιουργούνται λόγω των αναδρομικών κανόνων:

%right **','**
%left **'<' '>' '=' "comparator: not equal" "comparator: less or equal" "comparator: greater or equal" "keyword: and" "keyword: or" "operator: AND" "operator: OR"**
%left **'-' '+'**
%left **'*' '/' KW_MOD**
%left "*negative*"

Το αποτέλεσμα της εργασίας μας δεν διασφαλίζει την επίλυση προβλημάτων που εμπίπτουν σε στάδια μεταγενέστερα της λεκτικής και συντακτικής ανάλυσης, όπως είναι η σημασιολογική ανάλυση. Για παράδειγμα, οι κανόνες μας δεν διασφαλίζουν την ύπαρξη συνάρτησης με το όνομα "main" στο πρόγραμμα, την συνέπεια των ονομάτων και των τιμών των μεταβλητών ανάλογα με τον τύπο τους, ή αν το αποτέλεσμα μιας έκφρασης που δίνει τις διαστάσεις ενός πίνακα είναι πάντα ακέραιο.