HY340 : ΓΛΩΣΣΕΣ ΚΑΙ ΜΕΤΑΦΡΑΣΤΕΣ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ,
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ,
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ


```
VAR i:Integer;
    FUNCTION(Symbol) replicate
        x = (function(x,y){return x+y;});
        class DelFunctor: public std::unary_function<
```


ΔΙΔΑΣΚΩΝ
Αντώνιος Σαββίδης

Φροντιστήριο 1ο

Εισαγωγή στο FLEX

Flex

- Μια γεννήτρια λεξικογραφικών αναλυτών για τις γλώσσες C/C++
- Για την περιγραφή του λεξικογραφικού αναλυτή χρησιμοποιούνται:
  - ***Regular expressions***, που περιγράφουν τα πρότυπα της γλώσσας
  - ***Actions***, δηλαδή ενέργειες που πρέπει να πραγματοποιηθούν όταν αναγνωριστεί κάποιο συγκεκριμένο regular expression

Regular Expressions (1/4)

- **x** – αναγνωρίζει το χαρακτήρα *x*
- **“abcd”** - αναγνωρίζει την ακολουθία *abcd*
- **.** (τελεία) - αναγνωρίζει οποιοδήποτε χαρακτήρα ή σύμβολο εκτός από το new line
- **[xyz]** – αναγνωρίζει ένα από τους χαρακτήρες μέσα στο σύνολο, δηλαδή το *x*, το *y* ή το *z*
- **[ab-eg]** – αναγνωρίζει τους χαρακτήρες *a*, *b* έως *e* και *g*
- **[^ab-e]** – αναγνωρίζει οποιοδήποτε χαρακτήρα ή σύμβολο δεν ανήκει στο σύνολο *[ab-e]* (δηλαδή οτιδήποτε εκτός από a, b, c, d, e)

Regular Expressions (2/4)

- **r*** - αναγνωρίζει καμία ή περισσότερες επαναλήψεις του r
  - a* περιγράφει τα ε, a, aa, aaa, …
- **r+** - αναγνωρίζει μια ή περισσότερες επαναλήψεις του r
  - a+ περιγράφει τα a, aa, aaa, …, αλλά όχι το ε
- **r?** - αναγνωρίζει καμία ή μια επανάληψη του r (διαβάζεται και προαιρετικό r)
- **r{i, j}** - αναγνωρίζει i έως j επαναλήψεις του r
  - (όπου i, j > 0 και i < j)
- **r{i}** – αναγνωρίζει ακριβώς i επαναλήψεις του r
- **r{i,}** – αναγνωρίζει i ή περισσότερες επαναλήψεις του r

Regular Expressions (3/4)

- **rs** – αναγνωρίζει τις ακολουθίες που αναγνωρίζει η συνένωση των *r* και *s* (concatenation)
- **(r)** – αναγνωρίζει την ακολουθία *r* (χρησιμοποιείται για να καθορίσει την προτεραιότητα)
  - abc+ αναγνωρίζει τα abc, abcc, abccc, …
  - (abc)+ αναγνωρίζει τα abc, abcabc, …
- **r|s** – αναγνωρίζει το *r* ή το *s*
- **^r** – αναγνωρίζει το *r* αλλά μόνο όταν βρίσκεται στην αρχή της γραμμής
- **r$** – αναγνωρίζει το *r* αλλά μόνο όταν βρίσκεται στο τέλος της γραμμής
- **\\, \", \(, \), *, \+, \[, \], \$, \^, \{, \}, ...** – αναγνωρίζει τους ίδιους τους χαρακτήρες \, “, (, *, … (escaped)

Regular Expressions (4/4)

- Προσοχή στην προτεραιότητα!!
  - Το ***foo|bar**** είναι ισοδύναμο με ***(foo)|(ba(r*))***
    - Ο τελεστής ‘*’ έχει μεγαλύτερη προτεραιότητα από την ακολουθία χαρακτήρων
    - Η ακολουθία χαρακτήρων έχει μεγαλύτερη προτεραιότητα από τον τελεστή ‘|’
  - Αν θέλαμε να αναγνωρίσουμε το foo ή μηδέν ή περισσότερα bar θα γράφαμε ***foo|(bar)****
  - Αν θέλαμε να αναγνωρίσουμε μηδέν ή περισσότερα foo ή bar θα γράφαμε ***(foo|bar)****

Δομή Προγράμματος Flex

Τμήμα ορισμών

%%

Τμήμα κανόνων

%%

Τμήμα κώδικα χρήστη (προαιρετικό)

Τμήμα Ορισμών - Γενικά

1. **Κώδικας Χρήστη**
   - Αντιγράφεται αυτούσιος στο παραγόμενο αρχείο με τον κώδικα του λεξικογραφικού αναλυτή
   - Πρέπει να βρίσκεται μέσα στα σύμβολα {% /*code*/ %} ή %top{ /*code*/ }
2. **Regular expression macros (aliases)**
   - Συντάσσονται *name regex* και βοηθούν στην αναγνωσιμότητα του προγράμματος
   - Π.χ. *string* αντί για \"[^\n"]*\"
3. **Παράμετροι για τον παραγόμενο λεξικογραφικό αναλυτή**
   - %option option_name ή %option option_name=value
4. **User-defined conditions**
   - Κανόνες που ενεργοποιούνται με βάση την κατάσταση στην οποία βρίσκεται ο λεξικογραφικός αναλυτής

Τμήμα Ορισμών – Παράμετροι (1/2)

- ***%option header-file="./scanner.h"***
  - Δημιουργεί ένα header file που περιέχει τις δηλώσεις για τους τύπους και τις συναρτήσεις που χρησιμοποιούνται από τον παραγόμενο λεξικογραφικό αναλυτή
- ***%option noyywrap***
  - Δεν χρησιμοποιεί τη συνάρτηση "yywrap", που καλείται από τον παραγόμενο λεξικογραφικό αναλυτή όταν τελειώσει το διάβασμα ενός αρχείου (αν αυτή επιστρέψει 0 ο αναλυτής συνεχίσει το scanning από την είσοδο)
- ***%option yylineno***
  - Δηλώνει μια καθολική μεταβλητή με όνομα "yylineno", που κρατάει τον αριθμό της τρέχουσας γραμμής του αρχείου εισόδου
- ***%option case-insensitive***
  - Ο παραγόμενος λεξικογραφικός αναλυτής είναι case insensitive. Αυτό σημαίνει ότι η λέξη "ClaSS" θα είναι ίδια με τη λέξη "class"

Τμήμα Ορισμών – Παράμετροι (2/2)

- ***%option reentrant***
  - Ο παραγόμενος λεξικογραφικός αναλυτής είναι reentrant (μπορούμε να κάνουμε νέα κλήση στην yylex πριν τελειώσει η προηγουμενη). Προσοχή καθώς αλλάζει ο τρόπος κλήσης της yylex
- ***%option prefix="PREFIX"***
  - Εξ' ορισμού οι τύποι και οι συναρτήσεις που χρησιμοποιεί ο παραγόμενος λεξικογραφικός αναλυτής έχουν το πρόθεμα "yy" πχ. yylex, yyin, yylineno, yytext, yywrap, ... Αυτή η παράμετρος αλλάζει το πρόθεμα σε "PREFIX"
- ***%option nounistd***
  - Δεν κάνει 'include' το header file "unistd.h", το οποίο υπάρχει μόνο στα UNIX συστήματα

Τμήμα Ορισμών – User defined conditions

- Οι κανόνες μπορούν να ενεργοποιούνται υπό συνθήκη μόνο όταν βρισκόμαστε σε συγκεκριμένες καταστάσεις
- Υπάρχουν δύο είδη conditions
  - **Inclusive (start) conditions**
    - Ορίζονται με την εντολή "%s condition_name"
    - Μπορούν να ενεργοποιούν τους κανόνες που δε βασίζονται σε καμία κατάσταση (είναι χωρίς condition) καθώς και αυτούς που βασίζονται στην κατάσταση "<condition_name>"
  - **Exclusive conditions**
    - Ορίζονται με την εντολή "%x condition_name"
    - Μπορούν να ενεργοποιήσουν μόνο τους κανόνες που βασίζονται στην κατάσταση "<condition_name>"

Τμήμα Ορισμών – Παράδειγμα

```
%{
#if defined(WIN32)
#define YY_NO_UNISTD_H
static int isatty (int i) { return 0; }
#elif defined(_WIN32_WCE)
#define YY_NO_UNISTD_H
static int isatty (void *i) { return 0; }
#endif

#include <stdlib.h>
#include "Expressions.h"
#include "Statements.h"
#include "util.h"
#include "DecisionMakerBisonParser.h"

#define YY_DECL int yylex (YYSTYPE* lvalp)
%}

%option prefix = "DMSL_yy"
%option outfile = "Src/DecisionMakerScanner.cpp"
%option noyywrap
%option yylineno

id                  [a-zA-Z][a-zA-Z_0-9]*
integer             (0[xX][0-9A-Fa-f]+)|([0-9]+)
real                [0-9]*\.[0-9](e|E)?[0-9]*
space               [\r\n \t\v]
string              \"[^\n"]*\"
comment1            "//".*
comment2            "#".*

%x COMMENT
%%
```

Κώδικας

Παράμετροι

Ορισμοί regular expressions

User defined conditions

Τμήμα Κανόνων - Γενικά

- Το κύριο τμήμα του προγράμματος περιγραφής του λεξικογραφικού αναλυτή, αφού σε αυτό καθορίζεται η λειτουργικότητά του
- Σύνταξη: ***<condition1, condition2, ...> Regular expression { action }***
- Όταν παραπάνω από ένας κανόνας ικανοποιείται, τότε επιλέγεται αυτός που καταναλώνει τους *περισσότερους χαρακτήρες*
- Αν βρεθούν δύο ή παραπάνω κανόνες που καταναλώνουν τον ίδιο αριθμό χαρακτήρων τότε επιλέγεται αυτός που έχει δηλωθεί *πρώτος*

Τμήμα Κανόνων - Conditions

- ***Conditions***
  - Τα conditions ανάμεσα στα “<”, “>” αναφέρονται σε conditions που έχει ορίσει ο χρήστης (με “%s” ή “%x”)
  - Η παράθεσή τους είναι προαιρετική. Αν παραληφθούν ο κανόνας μπορεί να ενεργοποιηθεί είτε από την default κατάσταση, είτε από τα inclusive conditions
  - Η default κατάσταση στην οποία βρίσκεται στην αρχή ο λεξικογραφικός αναλυτής (και η οποία είναι inclusive), είναι η *INITIAL* και μπορεί να συμμετέχει κανονικά στη λίστα των conditions που προηγείται της κανονικής έκφρασης
  - Η κατάσταση “<*>” είναι συντομογραφία για **όλες** τις καταστάσεις που έχει δηλώσει ο χρήστης και για την *INITIAL*

Τμήμα Κανόνων – Regular Expressions & Actions

- ***Regular Expression***
  - Το regular expression μπορεί να περιλαμβάνει ή να αποτελείται αποκλειστικά από τα macros που έχουν δηλωθεί στο τμήμα ορισμών, τα οποία θα πρέπει να περικλείονται στα σύμβολα “{“, “}”, π.χ. {comment}
  - Το ειδικό σύμβολο “<<EOF>>” ικανοποιείται όταν τελειώσει η ανάγνωση του τρέχοντος αρχείου
- ***Action***
  - Ενεργοποιείται όταν ικανοποιείτε κάποια από τις καταστάσεις **και** το regular expression που ακολουθεί
  - Αν ο κώδικάς του είναι πάνω από ένα statement θα πρέπει να περικλείεται σε {, } (μέσα σε block)
  - Ο κώδικας μπορεί να χρησιμοποιεί κάποιες συναρτήσεις και μεταβλητές του παραγόμενου λεξικογραφικού αναλυτή

Τμήμα Κανόνων – Διαθέσιμες μεταβλητές και συναρτήσεις

- ***yytext:*** pointer σε char (ή char array στον κλασσικό lex) που περιέχει το κομμάτι του κειμένου που έχει ικανοποιήσει το regular expression
- ***yyleng*****:** ακέραιος που δηλώνει το μέγεθος του yytext
- ***BEGIN(condition):*** βάζει το λεξικογραφικό αναλυτή να μεταβεί στην κατάσταση με όνομα *"condition"*
- ***ECHO:*** αντιγράφει τα περιεχόμενα του *yytext στην έξοδο.*
- ***REJECT:*** βάζει το λεξικογραφικό αναλυτή να ενεργοποιήσει το «δεύτερο καλύτερο» κανόνα
  - ***Προσοχή:*** *Η χρήση του REJECT σε οποιονδήποτε κανόνα κάνει πολύ πιο αργό ολόκληρο το* λεξικογραφικό αναλυτή
- ***input():*** διαβάζει και επιστρέφει τον επόμενο χαρακτήρα από την ακολουθία εισόδου *(look-ahead)*
- ***unput(c):*** τοποθετεί το χαρακτήρα c στην ακολουθία εισόδου. Έτσι ο επόμενος χαρακτήρας που θα διαβάσει ο λεξικογραφικός αναλυτής θα είναι ο c

Τμήμα Κανόνων - Παραδείγματα

```
%{
#include <string.h>
%}

alpha . . .    [a-zA-Z]
comment1. .    "//".*

%option noyywrap

%%
"/*" . . .     {
. . . . . .        int c;
. . . . . .        while ((c = input()) != EOF) {
. . . . . .            if (c == '*') {
. . . . . .            .    if ((c = input()) == '/')
. . . . . .            .        break;
. . . . . .            .    else
. . . . . .            .        uput(c);
. . . . . .            }
. . . . . .        }
(comment1). .  {}
```

Κανόνες χωρίς condition

Κανόνες με condition

```
%{
#ifdef WIN32
#define YY_NO_UNISTD_H
int isatty (int i) { return 0; }
#endif

#include "Common.h"
#include "ConfigParser.h" // Only for YYSTYPE deps.
#include "Parser.h"

#include <stdlib.h>
#include <string.h>

#define YY_DECL int yylex (YYSTYPE* lvalp, Preprocessor* preproc)
%}

%option header-file="Configuration/Include/Scanner.h"
%option noyywrap
%option yylineno

id. . .    [a-zA-Z][a-zA-Z_0-9]*
space. .   [\r \t\v]
newline. . \n
string. .  \"[^\n"]*\"

comment1.  "//".*
comment2.  "#".*

%x COMMENT INCLUDE

%%
"/*"                BEGIN(COMMENT);
<COMMENT>[^*]*      /* eat anything that's not a '*' */
<COMMENT>"*"+[^*/]* /* eat up '*'s not followed by '/'s */
<COMMENT>"*"+"/"    BEGIN(INITIAL);
```


Τμήμα Κώδικα Χρήστη

- Το τμήμα κώδικα χρήστη είναι προαιρετικό και όταν παραλειφθεί μπορεί να παραλειφθεί και το δεύτερο σύμβολο "%%"
- Σκοπός του είναι μόνο η εύκολη και άμεση προσθήκη υλοποιήσεων των συναρτήσεων που χρησιμοποιούνται από τον παραγόμενο λεξικογραφικό αναλυτή
- Ό,τι προστίθεται σε αυτό το τμήμα αντιγράφεται χωρίς αλλαγές στο παραγόμενο αρχείο .c που περιέχει τον κώδικα του λεξικογραφικού αναλυτή
- Παράδειγμα:

```
%%
int main (int argc, char** argv)
{
    if (argc > 1) {
        if (!(yyin = fopen(argv[1], "r"))) {
            fprintf(stderr, "Cannot read file: %s\n", argv[1]);
            return 1;
        }
    }
    else
        yyin = stdin;

    yylex();
    return 0;
}
```

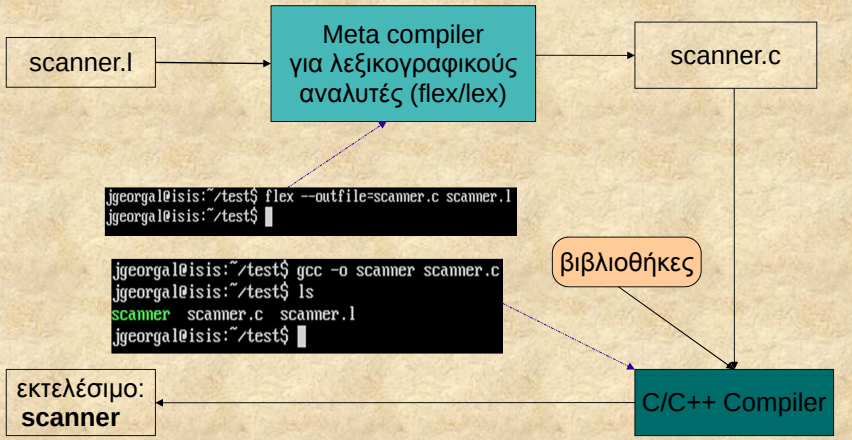

Ένας ολοκληρωμένος λεξικογραφικός αναλυτής

αρχείο scanner.l

```
/* Flex options */
%option noyywrap

/* Flex macros */
id. . .    [a-zA-Z][a-zA-Z_0-9]*
string. .  \"[^\n"]*\"
comment.   "//".*

%%

(id).      { fprintf(stderr, "Recognized id with value: %s\n", yytext); }
(string).  { fprintf(stderr, "Recognized string with value: %s\n", yytext); }
(comment). { fprintf(stderr, "Recognized comment with value: %s\n", yytext); }

. . .      { fprintf(stderr, "Cannot match character `%s' with any rule\n", yytext); }
%%

int main (int argc, char** argv)
{
    if (argc > 1) {
        if (!(yyin = fopen(argv[1], "r"))) {
            fprintf(stderr, "Cannot read file: %s\n", argv[1]);
            return 1;
        }
    }
    else
        yyin = stdin;

    yylex();
    return 0;
}
```

Ο παραγόμενος λεξικογραφικός αναλυτής, διαβάζει την ακολουθία χαρακτήρων εξ' ορισμού από το global FILE* pointer με όνομα "yyin".

Διαδικασία για την παραγωγή του τελικού προγράμματος



References

- Flex Home Page
  - http://www.gnu.org/software/flex
- Flex Manual
  - http://www.gnu.org/software/flex/manual/
- Flex for Windows
  - http://gnuwin32.sourceforge.net/packages/flex.htm