Διάλεξη 2η

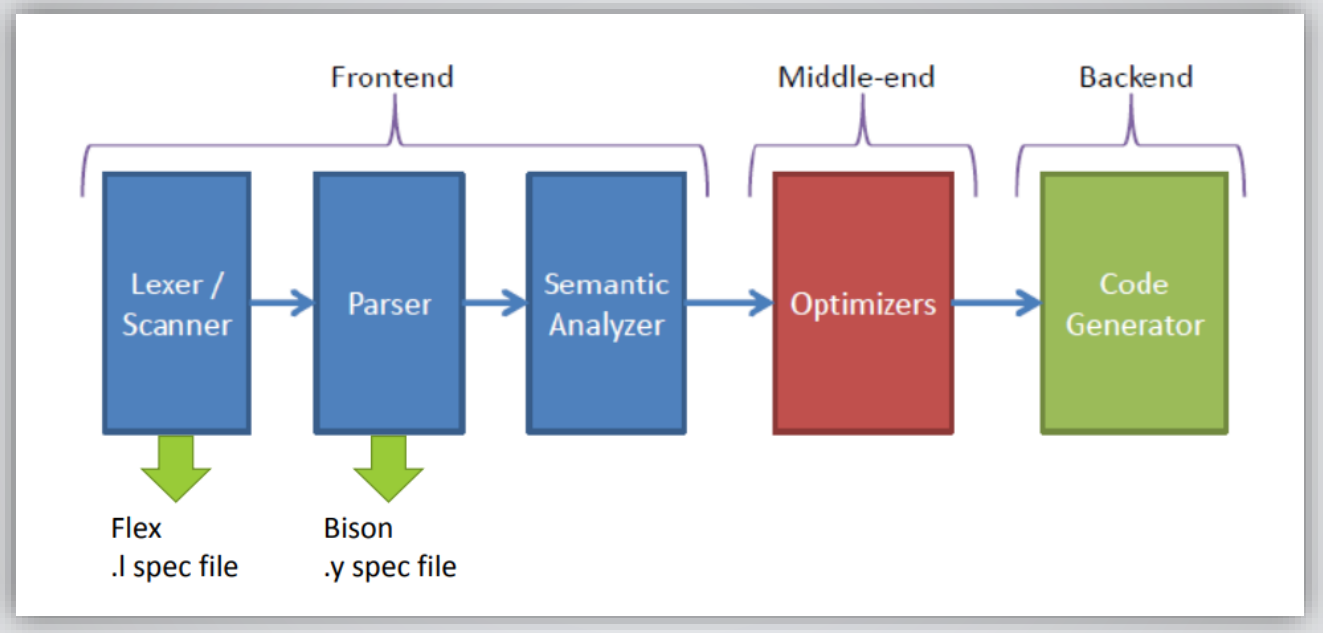
Εισαγωγή στο FLEX

Δομή και ένα απλό πρόγραμμα

Δημητριάδης Βασίλης
vdimitriadis@uth.gr

Αργυρίου Ιωάννης
iargyriou@uth.gr

Δομή/Flowchart ενός Μεταγλωττιστή



Τι είναι το FLEX;

- Εργαλείο Λεκτικής Ανάλυσης
- Ψάχνει και αναγνωρίζει τις **Λεκτικές Μονάδες** ενός προγράμματος
- **Επιστρέφει τιμές** (αριθμούς) για κάθε λεκτική μονάδα.

Input:

```
int my_number = 2 * 3 – 1;
```

tokens.h

T_INT	1
T_ID	2
T_ASSIGN	3
T_ICONST	4
T_MUL	5
T_MINUS	6
T_SEMI	7

Λεκτική Μονάδα	Τύπος	Token number
int	INTEGER	1
my_number	IDENTIFIER	2
=	ASSINGMENT OPERATOR	3
2	NUMBER	4
*	MULTIPLICATION OPERATOR	5
3	NUMBER	4
-	SUBSTRACTION OPERATOR	6
1	NUMBER	4
;	SEMI COLON	7

Τι είναι **Λεκτική Μονάδα**;

- **Λέξεις κλειδιά** – for, while, do
- **Αναγνωριστικά/Ονόματα** – my_number, x, iff
- **Σταθερές χωρίς πρόσημο** – 2, 2.3, 0.1e3, ‘a’
- **Τελεστές** – +, *, /, <, >, =
- **Διαχωριστές** – [,], (, ., ::, ;
- **Τέλος αρχείου** (EOF)

Λέξεις ΛΑ που δεν είναι λεκτικές μονάδες

- **Σχόλια** – // This is a comment
- **Κενά** – <<space>>, <<tab>>

Βρείτε τις Λεκτικές Μονάδες

```
1   typedef char ar[100];
2   enum days { monday=1, tuesday, wednesday, thursday, friday };
3   class t {
4   private:
5      float a,b; // double a,b;
6      int list al;
7   public:
8      float zz(int,ar);
9      union {
10        int i[2];
11        float f[2];
12     };
13  };
```

Εγκατάσταση **FLEX/BISON**

- **Windows**:
  - **Win Flex/Bison** (Easy): https://sourceforge.net/projects/winflexbison/
  - με **Virtual Machine** → Linux (Easy)
  - **CygWin** (Intermediate)
  - **Windows Linux Subsystem** (Intermediate)
- **Linux**:
  - sudo apt-get update
  - sudo apt-get upgrade
  - sudo apt-get install flex bison
- **Mac**:
  - http://macappstore.org/flex/
  - http://macappstore.org/bison/

Δομή FLEX (1)

Δηλώσεις (Βιβλιοθήκες, Ρυθμίσεις, Δηλώσεις συναρτήσεων, Classes Regex, κλπ...)
Κανόνες (RegEx, Επιστροφή τιμών, Εύρεση και Διόρθωση λαθών, κλπ..)
Συναρτήσεις Χρήστη (Main, Συνάρτηση Εκτύπωσης, Συνάρτηση Λάθους, κλπ..)

Δομή FLEX (2)

lexer.l

```
%{
        Κώδικας C
        (Βιβλιοθήκες, Μεταβλητές, Δηλώσεις Συναρτήσεων)
%}
Ρυθμίσεις FLEX
```

```
"IF"        { printf("Found IF\n"); return 1; }
"BILLY"     { printf("Found BILLY\n"); return 2; }
[a-zA-Z]*   { printf("Found random word"); return 3; }

.           { printf("Unknown\n"); throw error; }
```

```
int main(int argc, char *argv[]) {
    ……
}
……
```

Δομή FLEX (3)

```
%{
   #include <stdio.h>
   #include <stdlib.h>
%}

%option noyywrap    ← Διάβασε μόνο 1 αρχείο
%option case-insensitive    ← Δεν έχει σημασία μεταξύ κεφαλαίων-πεζών γραμμάτων

%%    ← Διαχωριστής

"IF"            { printf("Found IF\n"); return 1; }
"BILLY"         { printf("Found BILLY\n"); return 2; }
[a-zA-Z]*       { printf("Found random word"); return 3; }
<<EOF>>         { return 0; }    ← End Of File. Επιστρέφει πάντα 0!
.               { printf("Unknown\n"); }

%%    ← Διαχωριστής

int main(int argc, char *argv[]) { }
```

Το Flex ψάχνει **από πάνω προς τα κάτω** τους κανόνες

Δομή FLEX (4)

```
int main(int argc, char *argv[]){
        int token;
        if(argc > 1){
                yyin = fopen(argv[1], "r");
                if (yyin == NULL){
                        perror ("Error opening file");
                        return -1;
                }
        }

        do{
                token = yylex();
        }while(token != 0);

        fclose(yyin);
        yyterminate();
}
```

Διάβασε το αρχείο προς μετάφραση

Διάβασε το token δες τι είναι (βάση RegEx), μέχρι να φτάσεις στο τέλος του αρχείου

Κλείσε το αρχείο και τον Λεκτικό αναλυτή

Εκτέλεση FLEX

Εκτελούμε κατά σειρά τις ακόλουθες εντολές:

1. **flex <ονομα αρχείου>.l** ← Κάνει compile το αρχείο FLEX που δημιουργήσαμε
2. **gcc lex.yy.c -lfl** ← Κάνει compile το νέο δημιουργημένο αρχείο
3. **./a.exe <<Test file>>** ← (Windows) Εκτέλεση Λεκτικού αναλυτή πάνω σε test αρχείο
   ./a <<Test file>> ← (Linux) Εκτέλεση Λεκτικού αναλυτή πάνω σε test αρχείο

Ενδέχεται οι εντολές ελαφρά να αλλάζουν
από λειτουργικό σε λειτουργικό

Ας το δούμε στην πράξη!

Live Coding

(ノ°□°)ノ ┻━┻

Δομή FLEX (5)

tokens.h

```
#define T_EOF    0
#define T_IF     1
#define T_BILLY  2
#define T_WORD   3
    .
    .
    .
```

1

lexer.l

```
%{
   #include “tokens.h”
   #include <stdio.h>
   #include <stdlib.h>
%}
.................

%%

“IF”          { printf(“Found IF\n”); return T_IF; }
“BILLY”       { printf(“Found BILLY\n”); return T_BILLY; }
[a-zA-Z]*     { printf(“Found random word”); return T_WORD; }
<<EOF>>       { return T_EOF; }
.             { printf(“Unknown\n”); }
                                          .
                                          .
                                          .
```

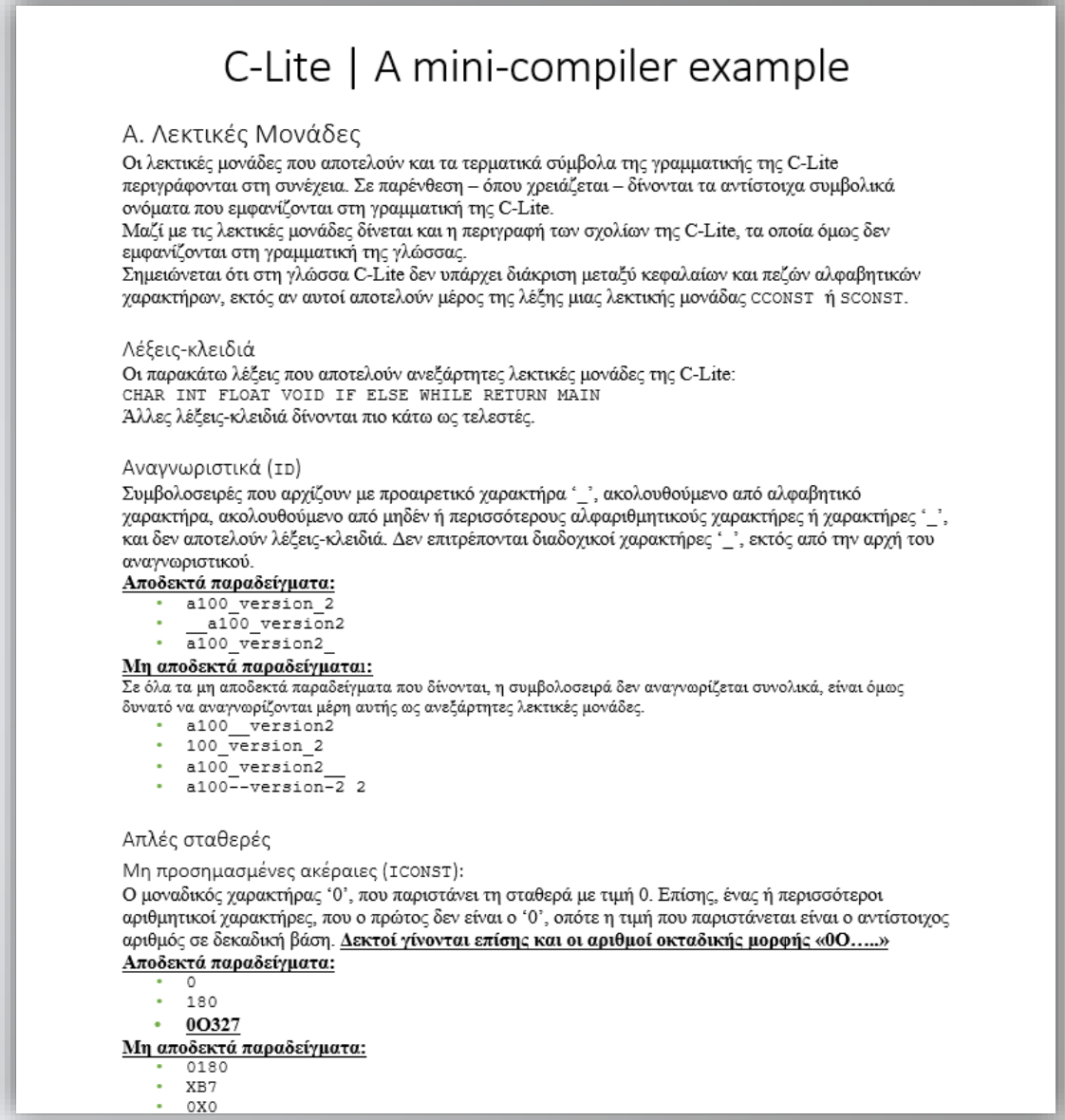
2

3

Φτιάχνουμε και το tokens.h
που θα χρειαστεί σε μελλοντικό στάδιο
(Bison)

Ο Μεταγλωττιστής **C-Lite**

- **Απλοποιημένη** μορφή μεταγλωττιστή
- **Βασικές Λέξεις Κλειδιά**
- Ακολουθεί την μορφή άσκησης που καλείστε να φτιάξετε
- Θα δουλέψουμε πάνω σε αυτόν τον μεταγλωττιστή στο εργαστήριο



C-Lite | A mini-compiler example

Α. Λεκτικές Μονάδες

Οι λεκτικές μονάδες που αποτελούν και τα τερματικά σύμβολα της γραμματικής της C-Lite περιγράφονται στη συνέχεια. Σε παρένθεση – όπου χρειάζεται – δίνονται τα αντίστοιχα συμβολικά ονόματα που εμφανίζονται στη γραμματική της C-Lite.
Μαζί με τις λεκτικές μονάδες δίνεται και η περιγραφή των σχολίων της C-Lite, τα οποία όμως δεν εμφανίζονται στη γραμματική της γλώσσας.
Σημειώνεται ότι στη γλώσσα C-Lite δεν υπάρχει διάκριση μεταξύ κεφαλαίων και πεζών αλφαβητικών χαρακτήρων, εκτός αν αυτοί αποτελούν μέρος της λέξης μιας λεκτικής μονάδας `CCONST` ή `SCONST`.

Λέξεις-κλειδιά

Οι παρακάτω λέξεις που αποτελούν ανεξάρτητες λεκτικές μονάδες της C-Lite:
`CHAR INT FLOAT VOID IF ELSE WHILE RETURN MAIN`
Άλλες λέξεις-κλειδιά δίνονται πιο κάτω ως τελεστές.

Αναγνωριστικά (`ID`)

Συμβολοσειρές που αρχίζουν με προαιρετικό χαρακτήρα '_', ακολουθούμενο από αλφαβητικό χαρακτήρα, ακολουθούμενο από μηδέν ή περισσότερους αλφαριθμητικούς χαρακτήρες ή χαρακτήρες '_', και δεν αποτελούν λέξεις-κλειδιά. Δεν επιτρέπονται διαδοχικοί χαρακτήρες '_', εκτός από την αρχή του αναγνωριστικού.

Αποδεκτά παραδείγματα:

- `a100_version_2`
- `__a100_version2`
- `a100_version2_`

Μη αποδεκτά παραδείγματα:

Σε όλα τα μη αποδεκτά παραδείγματα που δίνονται, η συμβολοσειρά δεν αναγνωρίζεται συνολικά, είναι όμως δυνατό να αναγνωρίζονται μέρη αυτής ως ανεξάρτητες λεκτικές μονάδες.

- `a100__version2`
- `100_version_2`
- `a100_version2__`
- `a100--version-2 2`

Απλές σταθερές

Μη προσημασμένες ακέραιες (`ICONST`):
Ο μοναδικός χαρακτήρας '0', που παριστάνει τη σταθερά με τιμή 0. Επίσης, ένας ή περισσότεροι αριθμητικοί χαρακτήρες, που ο πρώτος δεν είναι ο '0', οπότε η τιμή που παριστάνεται είναι ο αντίστοιχος αριθμός σε δεκαδική βάση. **Δεκτοί γίνονται επίσης και οι αριθμοί οκταδικής μορφής «0O…..»**

Αποδεκτά παραδείγματα:

- `0`
- `180`
- **`0O327`**

Μη αποδεκτά παραδείγματα:

- `0180`
- `XB7`
- `0X0`

Βοηθητικά options FLEX

%option noyywrap ← Διάβασε μόνο 1 αρχείο
%option case-insensitive ← Δεν έχει σημασία μεταξύ κεφαλαίων-πεζών γραμμάτων
%option yylineno ← Αφήνουμε το Flex να μετράει την γραμμή στην οποία είμαστε

Βοηθητικές συναρτήσεις FLEX

int **yylex()** ← Αναγνώρηση της επόμενης λεκτικής μονάδας.
void **yymore()** ← Ενσωματώνει την επόμενη λεκτική μονάδα στην τρέχουσα.
void **yyless(int n)** ← Κρατάει τους n χαρακτήρες του λεκτικού και επιστρέφει τους υπόλοιπους.
void **yyerror**(char* message) ← Εκτελούμε την συνάρτηση όταν υπάρχει λάθος

Βοηθητικές μεταβλητές FLEX

int **yylineno** ← Δείχνει την γραμμή (αν είναι το option yylineno ενεργοποιημένο)
char* **yytext** ← Δείχνει την παρούσα αναγνωρισμένη λέξη
int **yyleng** ← Δείχνει το μήκος της αναγνωρισμένης λέξης

Καλή συνέχεια

┬─┬ノ(゜-゜ノ)