

# Αρχές Γλωσσών Προγραμματισμού και Μεταφραστών: Αναφορά Εργαστηριακής Άσκησης 2012-2013

## Θεωρητικό Τμήμα

1. Για την παρακάτω γραμματική:

$$\begin{aligned}\langle S \rangle &::= a \langle A \rangle \mid b \mid \langle B \rangle \\ \langle A \rangle &::= \langle C \rangle a \mid \langle D \rangle b \\ \langle B \rangle &::= \langle C \rangle b \mid \langle D \rangle a \\ \langle C \rangle &::= \langle E \rangle \\ \langle D \rangle &::= \langle E \rangle \\ \langle E \rangle &::= \epsilon\end{aligned}$$

Τα σύνολα FIRST, FOLLOW και PREDICT είναι τα εξής:

- Σύνολα FIRST:  
 $\text{First}(S) = \{ a, b, \epsilon \}$   
 $\text{First}(A) = \{ \epsilon \}$   
 $\text{First}(B) = \{ \epsilon \}$   
 $\text{First}(C) = \{ \epsilon \}$   
 $\text{First}(D) = \{ \epsilon \}$   
 $\text{First}(E) = \{ \epsilon \}$
- Σύνολα FOLLOW:  
 $\text{Follow}(S) = \{ \$ \}$   
 $\text{Follow}(A) = \{ \$ \}$   
 $\text{Follow}(B) = \{ \$ \}$   
 $\text{Follow}(C) = \{ a, b \}$   
 $\text{Follow}(D) = \{ a, b \}$   
 $\text{Follow}(E) = \{ a, b \}$
- Σύνολα PREDICT:

1.  $\langle S \rangle ::= a \langle A \rangle$
2.  $\langle S \rangle ::= b$
3.  $\langle S \rangle ::= \langle B \rangle$
4.  $\langle A \rangle ::= \langle C \rangle a$
5.  $\langle A \rangle ::= \langle D \rangle b$
6.  $\langle B \rangle ::= \langle C \rangle b$
7.  $\langle B \rangle ::= \langle D \rangle a$
8.  $\langle C \rangle ::= \langle E \rangle$
9.  $\langle D \rangle ::= \langle E \rangle$
10.  $\langle E \rangle ::= \epsilon$

$$\begin{aligned}\text{Predict}(1) &= \{ a \} \\ \text{Predict}(2) &= \{ b \} \\ \text{Predict}(3) &= \{ \$ \} \\ \text{Predict}(4) &= \{ a, b \} \\ \text{Predict}(5) &= \{ a, b \} \\ \text{Predict}(6) &= \{ a, b \} \\ \text{Predict}(7) &= \{ a, b \}\end{aligned}$$

Predict(8)={ a,b }  
 Predict(9)={ a,b }  
 Predict(10)={ a,b }

### Πίνακας συντακτικής ανάλυσης

	a	b	\$\$
S	a <A> (1)	b (2)	<B> (3)
A	<C> a (4), <D> b (5)	<C> a (4), <D> b (5)	-
B	<C> b (6) , <D> a (7)	<C> b (6) , <D> a (7)	-
C	<E> (8)	<E> (8)	-
D	<E> (9)	<E> (9)	-
E	ε (10)	ε (10)	-

**Πίνακας 1**

Παρατηρείται ότι για τον κανόνα A με είσοδο a ο συντακτικός αναλυτής δε θα γνωρίζει ποιόν από τους κανόνες <C> a (4) ή <D> b (5) θα πρέπει να ακολουθήσει. Αυτό συμβαίνει στον κανόνα A με είσοδο b αλλά και στον κανόνα B με είσοδο a και b, αφού τα σύνολα predict των κανόνων 4 και 5, οι οποίοι έχουν ίδια αριστερά μέλη, ταυτίζονται. Το ίδιο συμβαίνει και στα σύνολα predict των κανόνων 6 και 7. Άρα η γλώσσα της δωθήσας γραμματικής δεν είναι LL(1)!

## **2.**

Στη συντακτική ανάλυση bottom-up υπάρχουν δύο επιλογές: ελάττωση με βάση έναν κανόνα και ολίσθηση στη στοίβα του επόμενου token εισόδου. Όταν σε κάποιο βήμα της συντακτικής ανάλυσης είναι αποδεκτές και οι δύο επιλογές τότε έχουμε σύγκρουση ολίσθησης-ελάττωσης. Οι συντακτικοί αναλυτές SLR επιλύουν το συγκεκριμένο πρόβλημα χρησιμοποιώντας τα σύνολα Follow, ενώ οι συντακτικοί αναλυτές LALR χρησιμοποιούν τοπική προανάγνωση.

### **ΧΑΡΑΚΤΗΡΙΣΤΙΚΗ ΜΗΧΑΝΗ ΠΕΠΕΡΑΣΜΕΝΗΣ ΚΑΤΑΣΤΑΣΗΣ**

Η γραμματική είναι:

S->A  
 A->yB  
 A->x  
 A->BC  
 B->zB  
 B->u  
 C->s

ΑΡΙΘΜΟΣ	ΚΑΤΑΣΤΑΣΗ	ΜΕΤΑΒΑΣΕΙΣ
0	<u>S-&gt;•A\$\$</u> A->•yB A->•x <u>A-&gt;•BC</u> B->•zB B->•u	Με A ολίσθηση και μετάβαση στην κατάσταση 4 Με y ολίσθηση και μετάβαση στην κατάσταση 1 Με x ολίσθηση και μετάβαση στην κατάσταση 5  Με z ολίσθηση και μετάβαση στην κατάσταση 7 Με u ολίσθηση και μετάβαση στην κατάσταση 2
1	<u>A-&gt;y•B</u> B->•zB B->•u	Με B ολίσθηση και μετάβαση στην κατάσταση 3 Με z ολίσθηση και μετάβαση στην κατάσταση 6 Με u ολίσθηση και μετάβαση στην κατάσταση 2
2	B->u•	Ελάττωση και αφαίρεση μίας κατάστασης
3	A->yB•	Ελάττωση και αφαίρεση δύο καταστάσεων
4	S->A•\$\$	Με \$\$ ολίσθηση και ελάττωση (αφαίρεση καταστάσεων και τοποθέτηση S στην είσοδο)
5	A->x•	Ελάττωση και αφαίρεση μίας κατάστασης
6	B->z•B B->•zB B->•u	Με z ολίσθηση και μετάβαση στην κατάσταση 6 Με u ολίσθηση και μετάβαση στην κατάσταση 2
7	A->•BC B->•zB B->•u	Με z ολίσθηση και μετάβαση στην κατάσταση 8 Με u ολίσθηση και μετάβαση στην κατάσταση 9
8	<u>B-&gt;z•B</u> B->•zB B->•u	Με B ολίσθηση και μετάβαση στην κατάσταση 10 Με z ολίσθηση και μετάβαση στην κατάσταση 8 Με u ολίσθηση και μετάβαση στην κατάσταση 9
9	B->u•	Ελάττωση και αφαίρεση μίας κατάστασης
10	<u>A-&gt;B•C</u> C->•s	Με C ολίσθηση και μετάβαση στην κατάσταση 12 Με s ολίσθηση και μετάβαση στην κατάσταση 11
11	C->s•	Ελάττωση και αφαίρεση μίας κατάστασης
12	A->BC•	Ελάττωση και αφαίρεση δύο καταστάσεων

**Πίνακας 2**

**Η BNF της γραμματικής μας είναι η εξής:**

<class> ::= CLASS KENO ID ANAG <variable2> KENO <constructor> KENO  
<methodos2> KLAG

<variable2> ::= <variable> | <variable2> KENO <variable>

<constructor> ::= <onoma> KENO <body>

<methodos2> ::= <methodos> | <methodos2> KENO <methodos>

<methodos> ::= <epistrofh> KENO <onoma> KENO <body>

<epistrofh>::=CHAR|INT|VOID  
 <onoma>::=ID ANPAR <orismata> KLPAR  
 <orismata>::=<orismaduo>|<orismata> KOMMA <orismaduo>|KENO  
 <orismaduo>::=<type> KENO <orismatria>  
 <orismatria>::=ID|<orismatria> ANSP KLSP  
 <type>::=CHAR|INT  
 <body>::=ANAG <entoles> KLAG|ANAG <entoles> KENO <return> KLAG  
 <entoles>::=<entolh>|<entoles> KENO <entolh>  
 <anathesh>::=<metavlhth> ISON <expr> ERWTHMATIKO  
 <elegxos>::=IF ANPAR <condition> KLPAR KENO <body> KENO ELSE KENO  
 <body> | WHILE ANPAR <condition> KLPAR KENO <body> |IF ANPAR  
 <condition> KLPAR KENO <body>  
 <expr>::=<oros> | <oros> SUN <expr>| <oros> PLIN <expr>  
 <oros>::=<paragontas> | <paragontas> EPI <oros> |<paragontas> DIA <oros>|  
 <paragontas> MOD <oros>  
 <paragontas>::=<metavlhth> | INTEGER |ANPAR <expr> KLPAR |CHARACTER  
 <metavlhth>::=ID | ID ANSP<expr> KLSP  
 <entolh>::=<anathesh>|<variable>|<elegxos>  
 <variable>::=DHLWSH | ARRAY | ANATHESH\_INT | ANATHESH\_CHAR  
 <return>::= RETURN KENO <expr> ERWTHMATIKO  
 <factor>::=ANPAR <sxesiakoi> KLPAR  
 <condition>::=<factor> KENO OR KENO <factor> | <factor> KENO AND KENO  
 <factor> |THAUM <factor>|<factor>  
 <sxesiakoi>::=<expr> EQUAL <expr>| <expr> NOT\_EQUAL <expr> |<praxh>  
 <praxh>::=<expr> LE <expr>|<expr> GE <expr>| <expr> ANST <expr>|<expr>  
 KLST <expr>

### Αρχείο Flex:

```
%x incl
%{
#include <stdio.h>
#include "y.tab.h"
#define MAX_INCLUDE_DEPTH 10
YY_BUFFER_STATE include_stack[MAX_INCLUDE_DEPTH];
int include_stack_ptr = 0;
void count ();
void comment ();
int column=0;
int lines=1;
%}

id [a-z_][a-zA-Z0-9_]*
digit [0-9]
character  \[a-zA-Z0-9. ? ; = * + _ - / ^ & > % { : )# | ~( ! , \ " ' < > ] \

%%
#"include" "      BEGIN(incl);
<incl>[ \t]*      /* eat the whitespace */
<incl>[^ \t\n]+ {
    if ( include_stack_ptr >= MAX_INCLUDE_DEPTH )
    {

        fprintf( stderr, "Includes nested too deeply" );
        exit( 1 );
    }

    include_stack[include_stack_ptr++] = YY_CURRENT_BUFFER;

    yyin = fopen( "file2.txt", "r" );

    yy_switch_to_buffer(
        yy_create_buffer( yyin, YY_BUF_SIZE ) );

    BEGIN(INITIAL);
}

<<EOF>> {
    if ( --include_stack_ptr < 0 )
    {
printf("End of file1");
yyterminate();
    }
}
```

```

[-]? {digit}+ {return (INTEGER); }
char {return CHAR; }
else { return ELSE; }
if { return IF; }
int {return INT; }
class { return CLASS; }
new { return NEW; }
return { return RETURN; }
void { return VOID; }
while {return WHILE; }
{id} { return ID; }
int" "{id}"="{digit}+";" {return ANATHESH_INT; }
char" "{id}"="{character}";" { return ANATHESH_CHAR; }
(char|int)" "{id}";" {return DHLWSH; }
{id}"="new" "(char|int)"["{digit}+]" "";" {return ARRAY; }
""""""""\0""""\t""""\{character} { return CHARACTER; }
[\n]+ {lines++; }
[\t]+ {}
"=" {return ISON; }
"/" {return DIA; }
"," {return ERWTHMATIKO; }
"+" {return SUN; }
"-" {return PLIN; }
"*" {return EPI; }
"%" {return MOD; }
"(" {return ANPAR; }
")" {return KLPAR; }
"[" {return ANSP; }
"]" {return KLSP; }
" " {return KENO; }
"{" {return ANAG; }
"}" {return KLAG; }
"," {return KOMMA; }
"!" {return THAUM; }
"<" {return ANST; }
">" {return KLST; }
"/*" { comment( ); }
"<=" {return LE; }

">=" {return GE; }

"==" {return EQUAL; }
"!=" {return NOT_EQUAL; }

```

```

"||" {return OR;}
"&&" {return AND;}
. { }

%%

void comment( )
{
    char c, prev = 0;

    while ((c = input()) != 0)    /* (EOF maps to 0) */
    {
        if (c == '/' && prev == '*')
            return;
        prev = c;
    }
    error("unterminated comment");
}

```

### **Αρχείο Bison:**

```

%{
#include "y.tab.h"
#include <stdio.h>
#include <stdlib.h>
void yyerror(char*);
extern FILE *yyin;
extern FILE *yyout;
extern int lines;
int errors=0;
%}

%token ID
%token INTEGER
%token CHARACTER
%token IF
%token ELSE
%token CLASS
%token NEW
%token RETURN
%token WHILE
%token VOID
%token INT
%token CHAR
%token ARRAY
%token ANATHESH_INT
%token ANATHESH_CHAR
%token DHLWSH
%token GE

```

```
%token LE
%token EQUAL
%token NOT_EQUAL
%token OR
%token AND
%token ISON
%token ERWTHMATIKO
```

```
%token SUN
%token PLIN
%token EPI
%token DIA
%token MOD
%token ANPAR
%token KLPAR
%token ANSP
%token KLSP
%token KENO
%token ANAG
%token KLAG
%token KOMMA
%token ANST
%token KLST
%token THAUM
```

```
%right '='
%left '<' '>'
%left '+' '-'
%left '%' '/' '*'
```

```
//%error-verbose
```

```
%% /*grammar rules*/
```

```
class: CLASS KENO ID ANAG variable2 KENO constructor KENO methodos2
KLAG|error KLAG {errors++; printf("Error class in line: %d \n", lines); yyclearin; };
```

```
variable2:variable|variable2 KENO variable;
```

```
constructor:onoma KENO body;
```

```
methodos2: methodos| methodos2 KENO methodos;
```

```
methodos:epistrofh KENO onoma KENO body;
```

```
epistrofh:CHAR|INT|VOID;
```

```
onoma:ID ANPAR orismata KLPAR|error KLPAR {errors++;printf("Error onoma in
line: %d \n", lines); yyclearin;};
```



```

orismata:orismaduo|orismata KOMMA orismaduo|KENO;

orismaduo:type KENO orismatria;

orismatria:ID|orismatria ANSP KLSP;

type:CHAR|INT;

body: ANAG entoles KLAG|ANAG entoles KENO return KLAG|error KLAG
{errors++; printf("Error body in line: %d \n", lines); yyclearin;};

entoles:entolh|entoles KENO entolh;

anathesh:metavlhth ISON expr ERWTHMATIKO|error ERWTHMATIKO {errors++;
printf("Error anathesh in line: %d \n", lines); yyclearin;};

elegxos:IF ANPAR condition KLPAR KENO body KENO ELSE KENO body |
WHILE ANPAR condition KLPAR KENO body |IF ANPAR condition KLPAR
KENO body| error KENO {errors++; printf("Error elegxos in line: %d \n", lines);
yyclearin; };

expr: oros | oros SUN expr| oros PLIN expr;

oros : paragontas | paragontas EPI oros |paragontas DIA oros|paragontas MOD oros ;

paragontas:metavlhth | INTEGER |ANPAR expr KLPAR |CHARACTER;

metavlhth:ID | ID ANSP expr KLSP ;

entolh:anathesh|variable|elegxos;

variable:DHLWSH | ARRAY | ANATHESH_INT | ANATHESH_CHAR;

return: RETURN KENO expr ERWTHMATIKO ;

factor: ANPAR sxesiakoi KLPAR|error KLPAR {errors++; printf("Error factor in line:
%d \n", lines); yyclearin;};

condition: factor KENO OR KENO factor | factor KENO AND KENO factor |
THAUM factor|factor;

sxesiakoi: expr EQUAL expr| expr NOT_EQUAL expr |praxh;

praxh: expr LE expr|expr GE expr| expr ANST expr|expr KLST expr;

%%

void yyerror(char* s)
{

```

```

fprintf(stderr, "%s\n", s);
}

int main(int argc, char **argv)
{
++argv;
--argc;
if(argc>0)
yyin=fopen(argv[0],"r");
else yyin=stdin;
yyout=fopen("output","w");
int a=yyparse();
if(a==0)
    printf("\nParsing ok...\n");
printf("Total number of errors in file: %d \n", errors);
printf("Total number of lines in file: %d\n", lines);
return 0;
}

```

### **Screenshots παραδειγμάτων εφαρμογής του psrser**

**(α)**

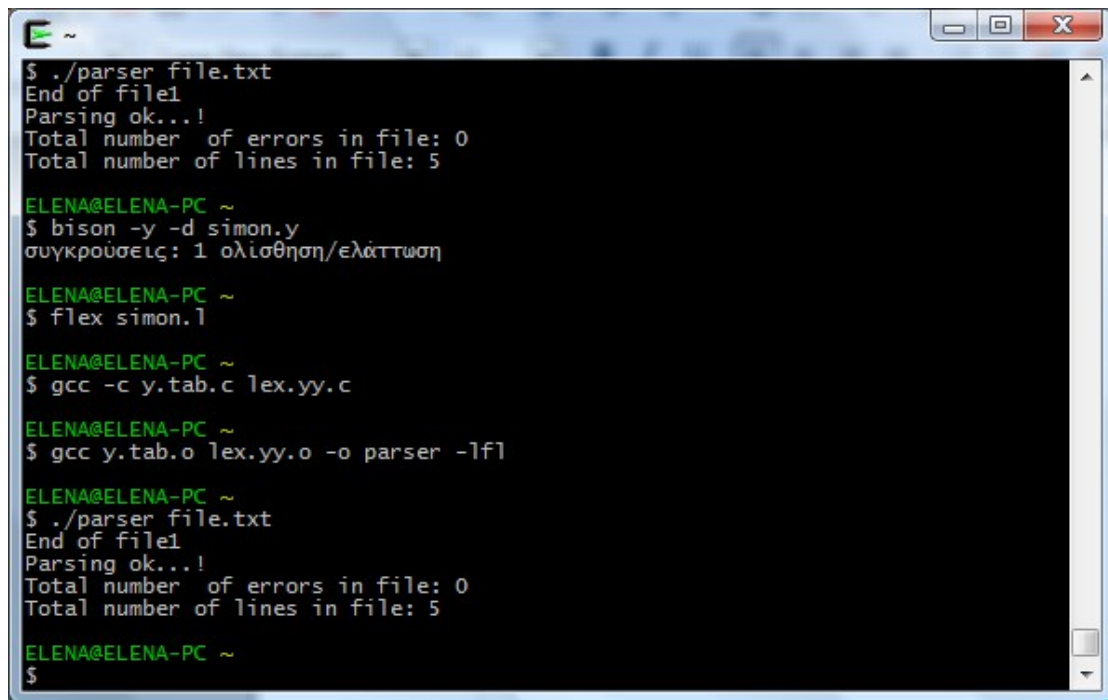
Με file1:

```

class cars {int x=5; name=new char[5];
car(char name[]) {if((x<=5)) {t=6;}}
/*xfgjfh*/
int engine(int k) {k=7; while((k<8)) {k=k+1;}}
}

```

## Screenshot:



```
$ ./parser file.txt
End of file1
Parsing ok...!
Total number of errors in file: 0
Total number of lines in file: 5

ELENA@ELENA-PC ~
$ bison -y -d simon.y
συγκρούσεις: 1 ολίσθηση/ελάττωση

ELENA@ELENA-PC ~
$ flex simon.l

ELENA@ELENA-PC ~
$ gcc -c y.tab.c lex.yy.c

ELENA@ELENA-PC ~
$ gcc y.tab.o lex.yy.o -o parser -lf1

ELENA@ELENA-PC ~
$ ./parser file.txt
End of file1
Parsing ok...!
Total number of errors in file: 0
Total number of lines in file: 5

ELENA@ELENA-PC ~
$
```

(β)

Με file1:

```
#include
```

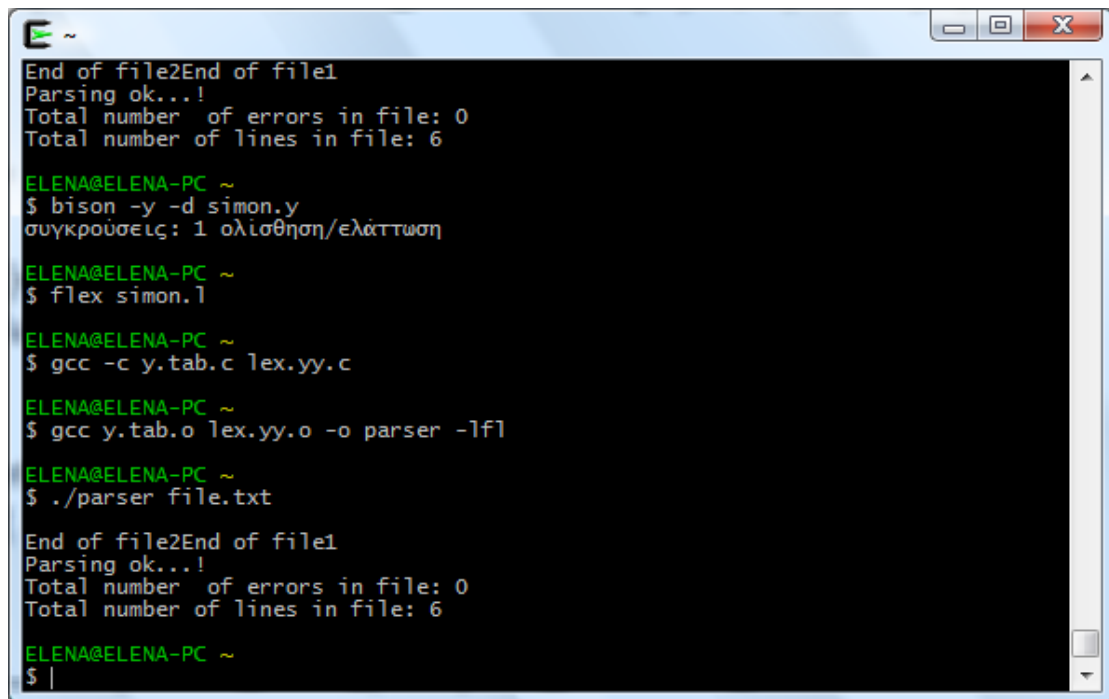
```
/*xfgjfh*/
```

```
int engine(int k) {k=7; while((k<8)) {k=k+1;}}
}
```

Με file2:

```
class cars{int x=5; name=new char[5];
car(char name[]) {if((x<=5)) {t=6;}}
```

Screenshot με το include:



```
~
End of file2End of file1
Parsing ok...!
Total number of errors in file: 0
Total number of lines in file: 6

ELENA@ELENA-PC ~
$ bison -y -d simon.y
συγκρούσεις: 1 ολίσθηση/ελάττωση

ELENA@ELENA-PC ~
$ flex simon.l

ELENA@ELENA-PC ~
$ gcc -c y.tab.c lex.yy.c

ELENA@ELENA-PC ~
$ gcc y.tab.o lex.yy.o -o parser -lf1

ELENA@ELENA-PC ~
$ ./parser file.txt

End of file2End of file1
Parsing ok...!
Total number of errors in file: 0
Total number of lines in file: 6

ELENA@ELENA-PC ~
$ |
```

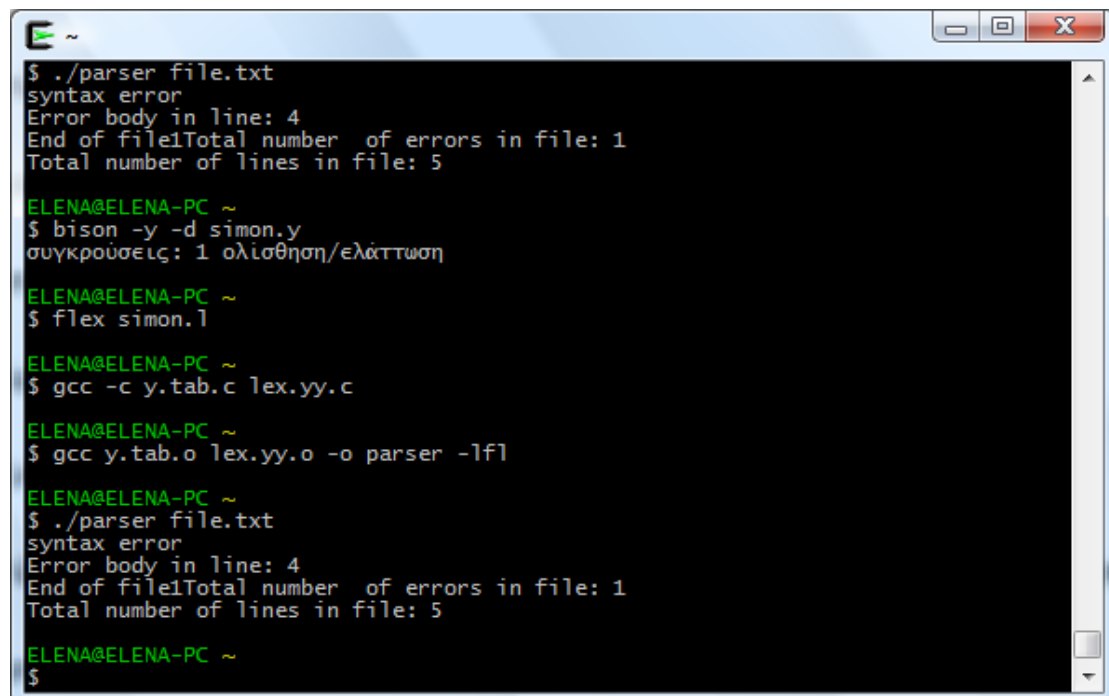
(γ)

Αφαιρούμε μια αγκύλη στο τέλος της γραμμής δύο μετά το `t=6;` άρα το αρχείο μας γίνεται ως εξής:

Με file1:

```
class cars{int x=5; name=new char[5];
car(char name[]) {if((x<=5)) {t=6;}
/*xfgjfh*/
int engine(int k) {k=7; while((k<8)) {k=k+1;}}
}
```

Screenshot με ένα error:



```
~
$ ./parser file.txt
syntax error
Error body in line: 4
End of file1Total number of errors in file: 1
Total number of lines in file: 5

ELENA@ELENA-PC ~
$ bison -y -d simon.y
συγκρούσεις: 1 ολίσθηση/ελάττωση

ELENA@ELENA-PC ~
$ flex simon.l

ELENA@ELENA-PC ~
$ gcc -c y.tab.c lex.yy.c

ELENA@ELENA-PC ~
$ gcc y.tab.o lex.yy.o -o parser -lf1

ELENA@ELENA-PC ~
$ ./parser file.txt
syntax error
Error body in line: 4
End of file1Total number of errors in file: 1
Total number of lines in file: 5

ELENA@ELENA-PC ~
$
```