

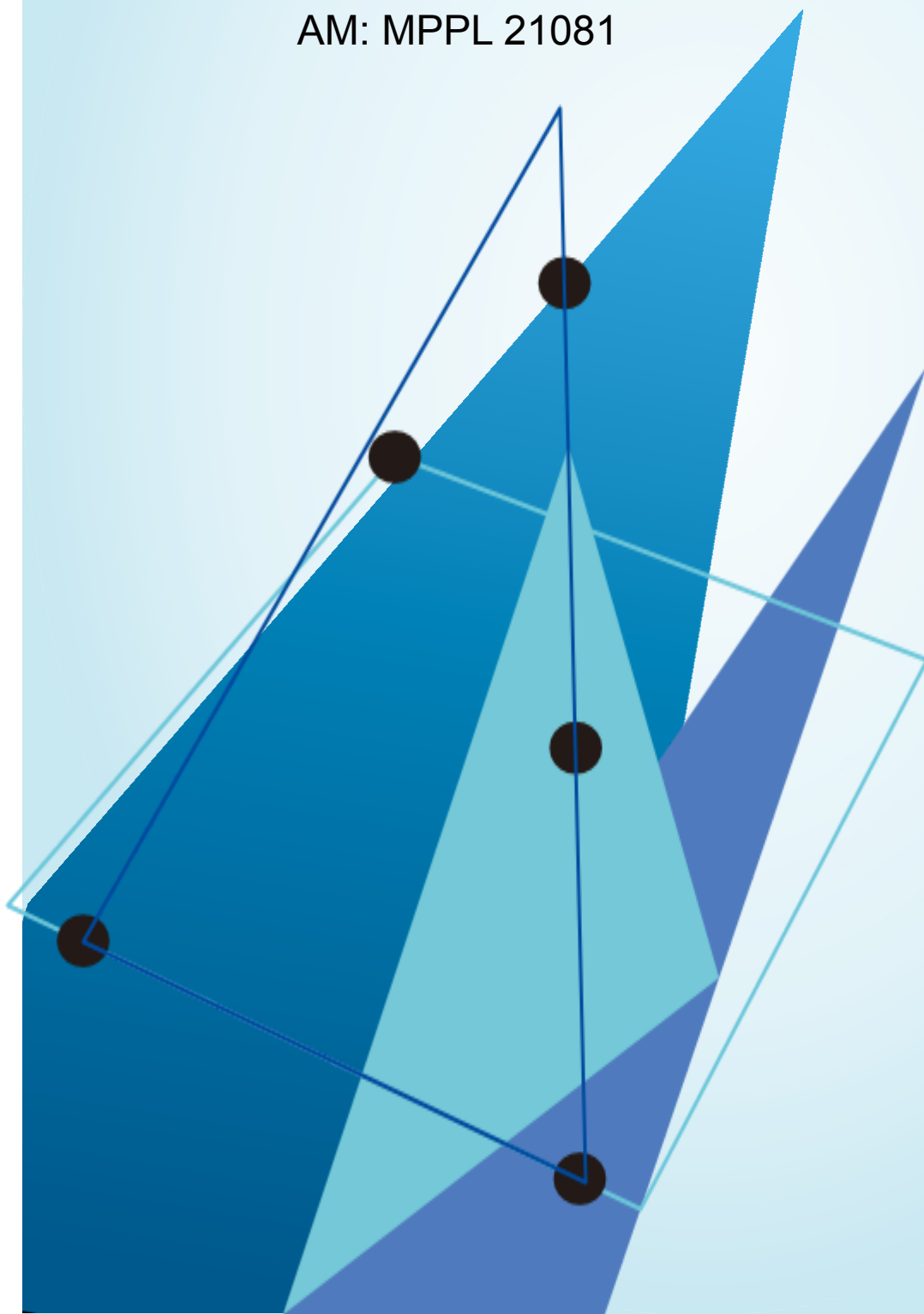
Τμήμα Πληροφορικής Πανεπιστήμιο πειραια

Πρόγραμμα Μεταπτυχιακών Σπουδών

Μάθημα : ΓΛΩΣΣΕΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΚΑΙ ΜΕΤΑΓΛΩΤΤΙΣΤΕΣ

ΧΡΟΝΟΠΟΥΛΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

ΑΜ: ΜΡΡΛ 21081



# ΘΕΜΑ Α

Με ΑΜ: mppl 21081 θα έχουμε ??1?? αντί για ??(?? ??8?? αντί για ??)?? . Το Ντετερμινιστικό Αυτόματο Στοιβάς έχει εκτελεστεί σε γλώσσα python και βρίσκεται στο αρχείο main.py , ο κώδικάς του όπτιου φαίνεται και στην παρακάτω εικόνα :

```
1 # deterministic pushdown automaton DPDA in Python
2 def main():
3     status = False
4     myStack = []
5     while status == False:
6         status = True
7         string = input("Please enter the expression for the DPDA : ")
8         for i in range(len(string)):
9             if string[i] != "1" and string[i] != "8":
10                 status = False
11                 print("The input has to be exclusively '1' or '8'")
12                 break
13     count = 0
14     for i in range(len(string)):
15         if string[i] == "1":
16             myStack.append(string[i])
17             print("Step " + str(i+1) + ".We push '1' to the stuck . Stuck : " + str(myStack) + "Rest of the expression : " + string[i:])
18             count = count + 1
19         elif string[i] == "8" and count > 0:
20             myStack.pop()
21             print("Step " + str(i+1) + ".We pop from the stuck . Stuck : " + str(myStack) + "Rest of the expression : " + string[i:])
22             count = count - 1
23         else:
24             print("NO the DPDA does not recognize the expression " + string)
25             exit()
26     if count == 0:
27         print("YES the DPDA does recognize the expression " + string)
28     else:
29         print("NO the DPDA does not recognize the expression " + string)
30
31 if __name__ == '__main__':
32     main()
```

Και εισάγοντας το “11181888” (που αντιστοιχεί στο “((()())”) παίρνουμε :

```
File Edit View Navigate Code Refactor Run Tools VCS Window Help pythonProject - main.py
pythonProject
main.py
Run: main
"D:\Python Pathakis tasks\pythonProject\venv\Scripts\python.exe" "D:/Python Pathakis tasks/pythonProject/main.py"
Please enter the expression for the DPDA : 11181888
Step 1 .We push '1' to the stuck . Stuck : ['1'] Rest of the expression : 11181888
Step 2 .We push '1' to the stuck . Stuck : ['1', '1'] Rest of the expression : 1181888
Step 3 .We push '1' to the stuck . Stuck : ['1', '1', '1'] Rest of the expression : 181888
Step 4 .We pop from the stuck . Stuck : ['1', '1', '1'] Rest of the expression : 81888
Step 5 .We push '1' to the stuck . Stuck : ['1', '1', '1', '1'] Rest of the expression : 1888
Step 6 .We pop from the stuck . Stuck : ['1', '1', '1', '1'] Rest of the expression : 888
Step 7 .We pop from the stuck . Stuck : ['1', '1', '1', '1'] Rest of the expression : 88
Step 8 .We pop from the stuck . Stuck : ['1', '1', '1', '1'] Rest of the expression : 8
YES the DPDA does recognize the expression 11181888
Process finished with exit code 0
```

## ΘΕΜΑ Β

Με ΑΜ : mppl 21081 έχουμε  $X = 1$ ,  $y = 8$ ,  $Z = 2$ . Άρα ένα παράδειγμα τμήματος δηλώσεων μεταβλητών που είναι έγκυρο για την γλώσσα που έχει περιγράψει είναι :

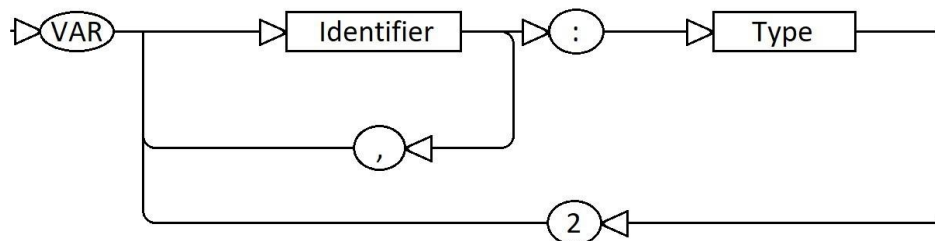
```
var num, sum , count : integer182
```

```
    tichet_price : real182
```

```
    fact_check : boolean182
```

```
    symbols : char182
```

Για το τμήμα δηλώσεων μεταβλητών που είναι έγκυρο για την γλώσσα που έχει περιγράψει Το συντακτικό διάγραμμα είναι :



Οπου Type μπορεί να είναι τα integer18 ,real18, boolean18, char18, Και identifier είναι το όνομά της μεταβλητής

Η περιγραφή EBNF είναι :

Variable-declaration-part =

```
var variable-declaration"2"{var variable-declaration"2"}
```

var variable-declaration =

```
identifier-list":type"18
```

identifier-list =

```
identifier{" , identifier}
```

identifier =

```
letter{letter|digit}
```

Οπου :

Type =

```
integer|real|boolean|char
```

Και :

Digit =

```
0|1|2|3|4|5|6|7|8|9
```

Letter =

```
a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
```

### Lex Πρόγραμμα :

Letter [A-Za-z]

Digit [0-9]

Identifier {Letter}{Letter|Digit}\*  
  
%%

"integer18"	{return INTEGER18;}
"real18"	{return REAL18;}
"boolean18"	{return BOOLEAN18;}
"char18"	{return CHAR18;}
[ ]	;
[,]	{return ",";}
[2]	{return 2;}
Identifier	{return Identifier;}
\n	{return \n;}
.	{return 0;}
%%	

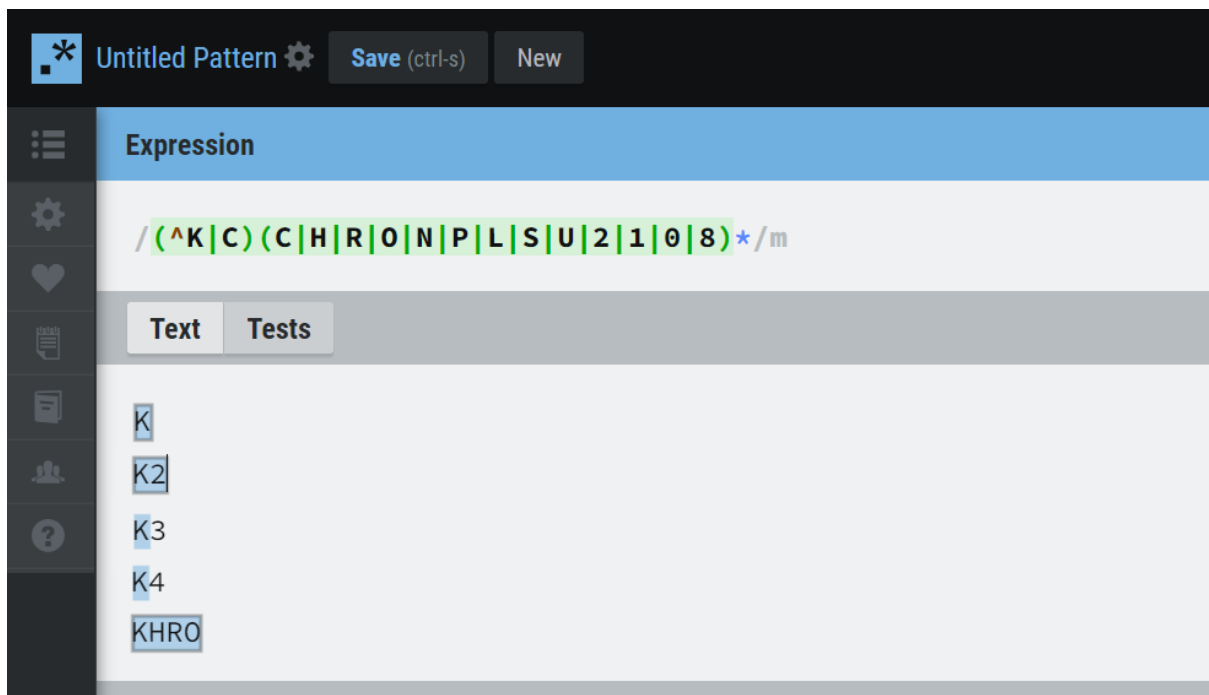
# ΘΕΜΑ Γ

Με Ονοματεπώνυμο : Chronopoulos Konstantinos και ΑΜ : μπρπ 21081 έχουμε Στη νέα αυτή γλωσσά μια μεταβλητή δηλώνεται ως εξής:

- Το αρχικό γράμμα θα είναι είτε Κ η C .
- Επειτα μπορεί να επαναλαμβάνονται οποιοδήποτε αριθμό φορές οι χαρακτήρες : “C” “H” “R” “O” “N” “P” “U” “L” “S” και “2” “1” “0” “8” .

Η κανονική έκφραση για τα ονόματα μεταβλητών θα είναι :

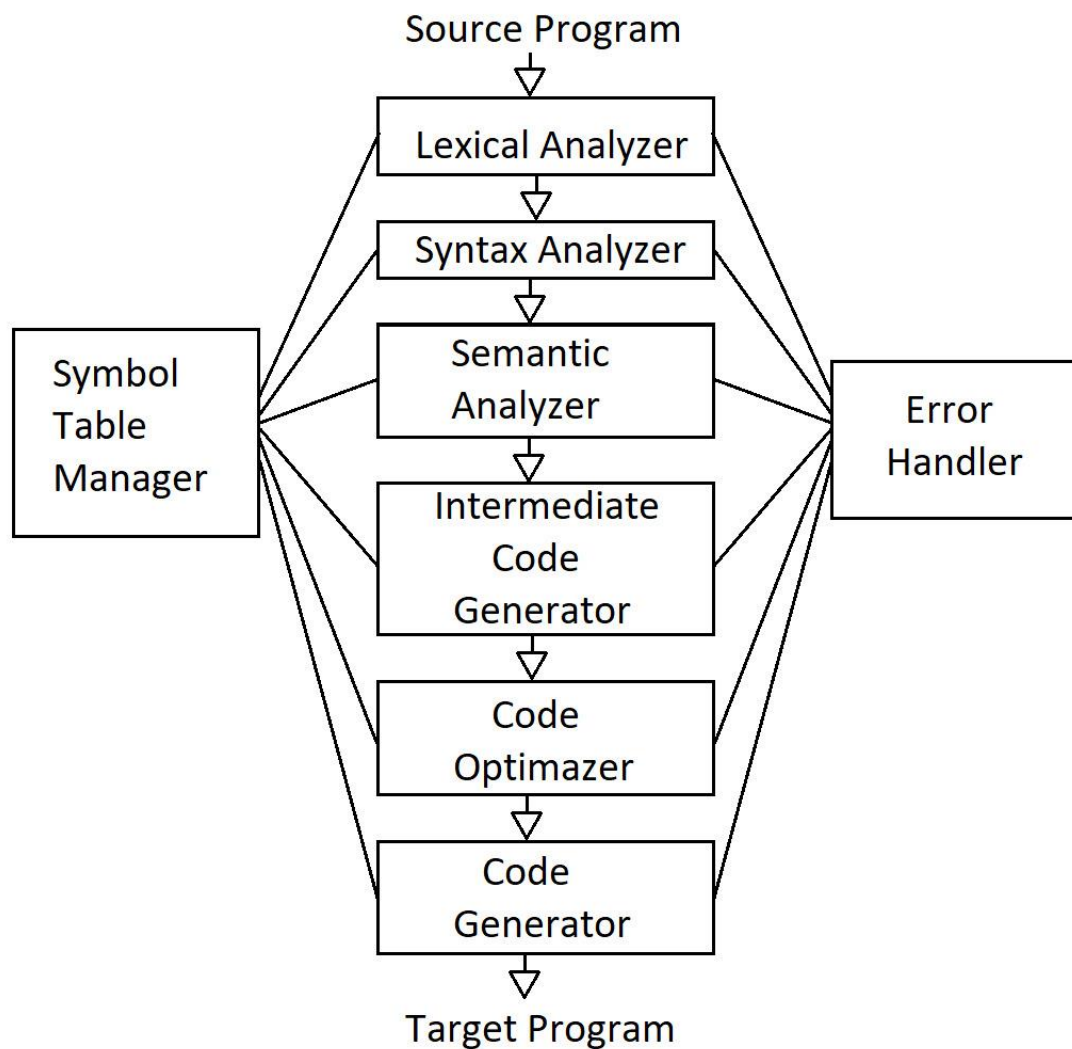
`/(^K|C)(C|H|R|O|N|P|L|S|U|2|1|0|8)* /`



## B)

Μεταγλωττιστή (compiler) ονομάζουμε ένα πρόγραμμα Ηλεκτρονικού Υπολογιστή που διαβάζει κώδικα γραμμένο σε μια γλώσσα προγραμματισμού (την πηγαία γλώσσα ) και τον μεταφράζει σε ισοδύναμο κώδικα σε μια άλλη γλώσσα προγραμματισμού(τη γλώσσα στόχο). Συνήθως ο όρος “μεταγλωττιστής” χρησιμοποιείται κυρίως για προγράμματα που μεταφράζουν μια γλώσσα προγραμματισμού υψηλού επιπέδου σε μια γλώσσα σε μια γλώσσα χαμηλού επιπέδου.

Στο παρακάτω διάγραμμα απεικονίζονται οι φάσεις που περνάει ο μεταγλωττιστής



Για την νέα γλώσσα προγραμματισμού μας δίνεται ο πηγαίος κώδικας :

```
int K = 2;
int K1 = 3;
int K2 = 4;
int K3 = 5;
int K4 = 4;
int K5 = 5;
int K6 = 4;
int K7 = 6;
int K8 = 7;
int K9 = 8;
if K==K1 {
    printf("K is equal to K1");
}
iff A > A1 {
    printf("K is greater than K1");
}
K1=K4;
K=K3;
K5=K;
K6=K1;
K7=K8;
K9=5;
```

### Λεκτικός Αναλυτής (lexical analyzer):

Ο Λεκτικός Αναλυτής είναι η πρώτη φάση του μεταγλωττιστή . Ο Λεξικός Αναλυτής παίρνει ως είσοδο τον πηγαίο κώδικα σαν συμβολοσειρά χαρακτήρων του προγράμματος και τον μετατρέπει σε μια συμβολοσειρά από tokens(λεκτικών μονάδων) και αφαιρεί τα επιπλέον κενά και σχόλια .Τα tokens αυτά συνήθως μπορεί να είναι 1. Λέξεις κλειδιά (keywords) 2. Τα αναγνωριστικά (identifiers) 3. Σταθερές (constants) 4. Οι τελεστές (operators).Μια σειρά χαρακτήρων η οποία δεν αναγνωρίζεται ως σωστό token είναι ένα λεξιλογικό λάθος πχ ο Λεκτικός αναλυτής (lexical analyzer) το "if" ως keyword αλλά δεν θα αναγνωρίσει το "iff". Το αποτέλεσμα λοιπόν του Λεκτικού αναλυτή θα είναι :

N	Lexeme	Type	N	Lexeme	Type	N	Lexeme	Type
1	int	keywords	22	K4	error	43	=	operator
2	K	identifier	23	=	operator	44	7	identifier
3	=	operator	24	4	identifier	45	;	special
4	2	identifier	25	;	special	46	int	keywords
5	;	special	26	int	keywords	47	K9	error
6	int	keywords	27	K5	error	48	=	operator
7	K1	identifier	28	=	operator	49	8	identifier
8	=	operator	29	5	identifier	50	;	special
9	3	identifier	30	;	special	51	if	keywords
10	;	special	31	int	keywords	52	K	identifier
12	int	keywords	32	K6	error	53	==	operator
12	K2	identifier	33	=	operator	54	K1	identifier
13	=	operator	34	4	identifier	55	{	operator
14	4	identifier	35	;	special	56	printf	identifier
15	;	special	36	int	keywords	57	(	operator
16	int	keywords	37	K7	error	58	"K is equal to K1"	identifier
17	K3	error	38	=	operator	59	)	operator
18	=	operator	39	6	identifier	60	;	special
19	5	identifier	40	;	special	61	}	operator
20	;	special	41	int	identifier	62	iff	error
21	int	keywords	42	K8	identifier	63	A	error

N	Lexeme	Type	N	Lexeme	Type	N	Lexeme	Type
64	>	operator	75	K4	error	86	=	operator
65	A1	error	76	;	special	87	K1	identifier
66	{	operator	77	K	identifier	88	;	special
67	printf	identifier	78	=	operator	89	K7	error
68	(	operator	79	K3	error	90	=	operator
69	"Kis greater than K1"	identifier	80	;	special	91	K8	identifier
70	)	operator	81	K5	error	92	;	special
71	;	special	82	=	operator	93	K9	error
72	}	operator	83	K	identifier	94	=	operator
73	K1	identifier	84	;	special	95	5	identifier
74	=	operator	85	K6	error	96	;	special

Αρά ο λεκτικός αναλυτής θα αναγνωρίσει τα K,K1,K2,K8 ως identifier αλλά τα K3,K4,K5, K6,K7,K9 και A , A1 δεν θα τα αναγνωρίζει ως identifier γιατί δεν τηρούν τις συνθήκες της γλώσσας.

Επίσης δεν θα αναγνωρίσει το "iff" ως keyword .

### **Συντακτικός Αναλυτής(syntax analyzer ):**

Ο Συντακτικός Αναλυτής είναι η δεύτερη φάση του μεταγλωττιστή ,θα πάρει ως είσοδο την παράπονο σειρά από tokens( token streams) και τα αναλύει με βάση μια σειρά από κανόνες παραγωγής για να αναγνωρίσει λαοί στον κώδικα .Ως έξοδο έχει ένα συντακτικό δένδρο (parse tree).

Ο Λεκτικός Αναλυτής αναγνωρίζει τα tokens με την βοήθεια κανονικών εκφράσεων (regular expressions) και κανόνων , ο Λεκτικός Αναλυτής όμως δεν μπορεί να ελέγξει την σύνταξη μιας πρότασης λόγω των περιορισμών των κανονικών εκφράσεων πχ οι κανονικές εκφράσεις δεν μπορούν να αναγνωρίσουν παρενθέσεις που ανοίγουν και κλείνουν ,Για Αυτό ο Συντακτικός αναλυτής χρησιμοποιεί context-free grammar που αναγνωρίζεται από τα push-down automata.

### **Σημασιολογικός Αναλυτής(Semantic Analyzer)**

Ο Σημασιολογικός Αναλυτής(Semantic Analyzer) είναι η τρίτη φάση του μεταγλωττιστή .ο σημασιολογικής ανάλυσης βεβαιώνει ότι οι δηλώσεις και εκφράσεις του προγράμματος είναι σημασιολογικά σούστες .Ο Σημασιολογικός Αναλυτής(Semantic



Analyzer) παίρνει ως είσοδο το συντακτικό δέντρο από την προηγούμενη φάση αλλά χρησιμοποιεί και τον πίνακα συμβολών (symbol table) για να δοκιμάσει αν το πρόγραμμα είναι Σημασιολογικά σωστό ,

### **Σημασιολογικά Λάθη :**

Παραδείγματα σημασιολογικών λαθών :

- Λάθος τύπος μεταβλητής.
- Μη δηλωμένες μεταβλητές.
- Λάθος χρήση δεσμευμένων identifier.