

Διάλεξη 5 - Τελεστές και Εντολές

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Εισαγωγή στον Προγραμματισμό

Θανάσης Αυγερινός / Τάκης Σταματόπουλος

Ανακοινώσεις / Διευκρινίσεις

- Εργασία 0: pages ερώτημα
 - a. Πρέπει να αποδεχτείτε την πρόσκληση *και* να φτιάξετε το δικό σας repository
 - b. Επομένως, υπάρχουν δύο repositories τα οποία θα δημιουργηθούν:
 - progintro/hw0-yourusername <- repository του μαθήματος
στο οποίο ανεβάζετε τις λύσεις σας για την εργασία
 - yourusername/[yourusername.github.io](https://github.com/yourusername/yourusername.github.io) <- προσωπικό repository στο οποίο μπορείτε να βάλετε το περιεχόμενο της ιστοσελίδας σας
- Coding Club: συνάντηση γνωριμίας στις 13:00 στην αίθουσα ΣΤ

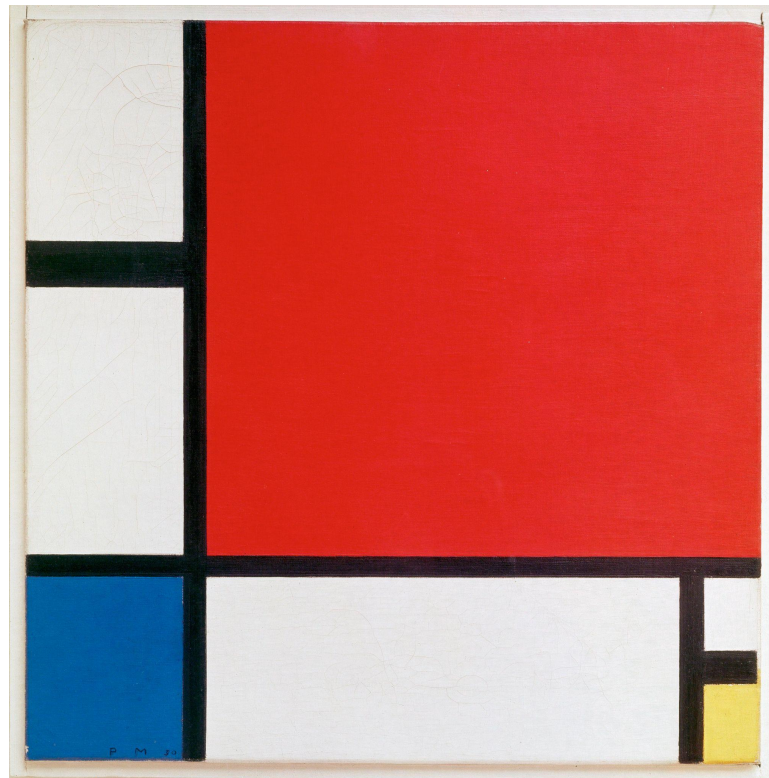
Την Προηγούμενη Φορά

- Pair Programming
- Git
- Τελεστές

GitHub

Σήμερα

- Τελεστές
- Εκφράσεις και Εντολές
- Ροή Ελέγχου (αρχικά)



Τελεστές (Operators)

- Μοναδιαίοι (unary), Δυαδικοί (binary), Τριαδικοί (ternary)
- Αριθμητικοί, Συγκριτικοί, Λογικοί, Συνθήκης, Bitwise, Μετατροπής, Ανάθεσης
- Προτεραιότητα & Προσεταιριστικότητα

Παραδείγματα Τελεστών

Αριθμητικοί Τελεστές

Αριθμητικός Τελεστής	Ερμηνεία	Παραδείγματα
+	πρόσθεση	$40 + 2$ επιστρέφει 42 $40.0 + 2.0$ επιστρέφει 42.0
-	αφαίρεση	$44 - 2$ επιστρέφει 42 $44.0 - 2.0$ επιστρέφει 42.0
*	πολλαπλασιασμός	$42 * 2$ επιστρέφει 84 $42.0 * 2.0$ επιστρέφει 84.0
/	διαίρεση	$85.0 / 2.0$ επιστρέφει 42.5 $85 / 2$ επιστρέφει 42
%	υπόλοιπο	$7 \% 2$ επιστρέφει 1

1. Σε τι κατηγορία θα εντάσσετε τους αριθμητικούς τελεστές;
2. Τι τύπο θα είχαν οι τελεστές αν ήταν συναρτήσεις;

Συγκριτικούς Τελεστές (Comparison Operators)

Οι **συγκριτικοί τελεστές** επιτρέπουν την σύγκριση τελεστών. Το αποτέλεσμα είναι **0** (ψευδές) ή **1** (αληθές). Τι τύπου τελεστές είναι;

Σχεσιακός Τελεστής	Ερμηνεία	Παραδείγματα
==	Ίσοι τελεστές;	42 == 0 επιστρέφει 0 42 == 42 επιστρέφει 1
!=	Διαφορετικοί τελεστές;	42 != 0 επιστρέφει 1 42 != 42 επιστρέφει 0
>	Αριστερός τελεστής μεγαλύτερος του δεξιού;	42 > 0 επιστρέφει 1 42 > 42 επιστρέφει 0
<	Αριστερός τελεστής μικρότερος του δεξιού;	42 < 0 επιστρέφει 0 42 < 44 επιστρέφει 1
>=	Αριστερός τελεστής μεγαλύτερος ή ίσος του δεξιού;	42 >= 42 επιστρέφει 1 42 >= 43 επιστρέφει 0
<=	Αριστερός τελεστής μικρότερος ή ίσος του δεξιού;	42 <= 42 επιστρέφει 1 42 <= 41 επιστρέφει 0

Λογικοί Τελεστές (Logical Operators)

Οι **λογικοί τελεστές** επιτρέπουν την αποτίμηση συνδυασμού αληθών και ψευδών τιμών. Τι τύπου τελεστές είναι;

Λογικός Τελεστής	Ερμηνεία	Παραδείγματα
&&	Λογική σύζευξη (AND, \wedge) είναι και οι δύο τελεστέοι αληθείς;	$42 > 0 \ \&\& \ 2 > 3$ επιστρέφει 0 $42 > 0 \ \&\& \ 3 > 2$ επιστρέφει 1
 	Λογική διάζευξη (OR, \vee) είναι ένας εκ των δύο τελεστέων αληθής;	$2 > 3 \ \ 42 > 0$ επιστρέφει 1 $2 > 3 \ \ 42 < 0$ επιστρέφει 0
!	Λογική άρνηση (NOT, \neg) είναι ο τελεστέος ψευδής;	$!(42 < 0)$ επιστρέφει 1 $!(42 > 0)$ επιστρέφει 0

Σημαντικό: για τους λογικούς τελεστές στην C, το μηδέν (0) σημαίνει “ψευδές”. Οποιαδήποτε άλλη μη μηδενική τιμή σημαίνει “αληθές”. Το 1, το 42 και το -5 είναι όλα εξίσου “αληθή”.

Bitwise Τελεστές (Bitwise Operators)

Οι **bitwise τελεστές** κάνουν μετατροπές στα *bit* των **ακέραιων** αριθμών. Τι τύπου είναι οι τελεστές;

Bitwise Τελεστής	Ερμηνεία	Παραδείγματα
&	Σύζευξη bit (bitwise AND)	0xFF & 0xF0 επιστρέφει 0xF0 0xFF & 0x00 επιστρέφει 0x00
	Διάζευξη bit (bitwise OR)	0xFF 0xF0 επιστρέφει 0xFF 0xF0 0x0F επιστρέφει 0xFF
^	Αποκλειστική διάζευξη bit (bitwise XOR)	0xFF ^ 0xFF επιστρέφει 0x00 0x00 ^ 0xFF επιστρέφει 0xFF
~	Άρνηση bit (bitwise NOT)	~0xF0 επιστρέφει 0x0F ~0x05 επιστρέφει 0xFA
<<	Ολίσθηση bit αριστερά (shift left). Shift N bit ισοδύναμο με πολλαπλασιασμό με 2^N	2 << 4 επιστρέφει 32 4 << 1 επιστρέφει 8
>>	Ολίσθηση bit δεξιά (shift right). Shift N bit ισοδύναμο με διαίρεση με 2^N	8 >> 1 επιστρέφει 4 32 >> 4 επιστρέφει 2

Πως απομονώνουμε το πιο σημαντικό bit σε ένα byte;

A. `byte & 0xFF`

B. `byte & 0x80`

C. `byte | 0xFF`

D. `byte | 0x80`

Ποια η τιμή της παράστασης $0xb\text{eef} \mid 0xc\text{afe}0000$;

- A. $0xffffffff$
- B. $0xc\text{afeb}\text{eef}$
- C. $0xb\text{eefc}\text{afe}$
- D. $0xf\text{aceb}00c$

Τελεστής Συνθήκης (Conditional Operator)

Ο τελεστής συνθήκης ελέγχει την συνθήκη και αν είναι αληθής επιστρέφει την πρώτη τιμή, αλλιώς επιστρέφει την δεύτερη. Γενική μορφή:

συνθήκη ? τιμή1 : τιμή2

Για παράδειγμα:

`(42 > 0) ? 8 : 16` => επιστρέφει 8

Αντίστοιχα:

`(42 == 0) ? 8 : 16` => επιστρέφει 16

Τι τύπου είναι ο τελεστής συνθήκης; Πως θα φτιάχναμε μια συνάρτηση max;

Τελεστής Μετατροπής (Cast Operator)

Ο τελεστής μετατροπής αλλάζει τον τύπο ενός τελεστέου. Γενική μορφή:

(τύπος)τελεστέος

Παραδείγματα:

(int)42.67 επιστρέφει 42

(char)67.8 επιστρέφει 'C'

(double)3 επιστρέφει 3.0

(double)3/4 επιστρέφει ??

Ψηφία μετά την
υποδιαστολή
αποκόπτονται, δεν
στρογγυλοποιούνται

Η παραπάνω μετατροπή λέγεται και explicit type conversion

Σιωπηρή Μετατροπή Τύπων (Implicit Type Conversion)

Σε περιπτώσεις μίξης τύπων σε έναν τελεστή, ο μεταγλωττιστής προσθέτει αυτόματα μετατροπές στον μεγαλύτερο των τελεστέων.

Παράδειγμα:

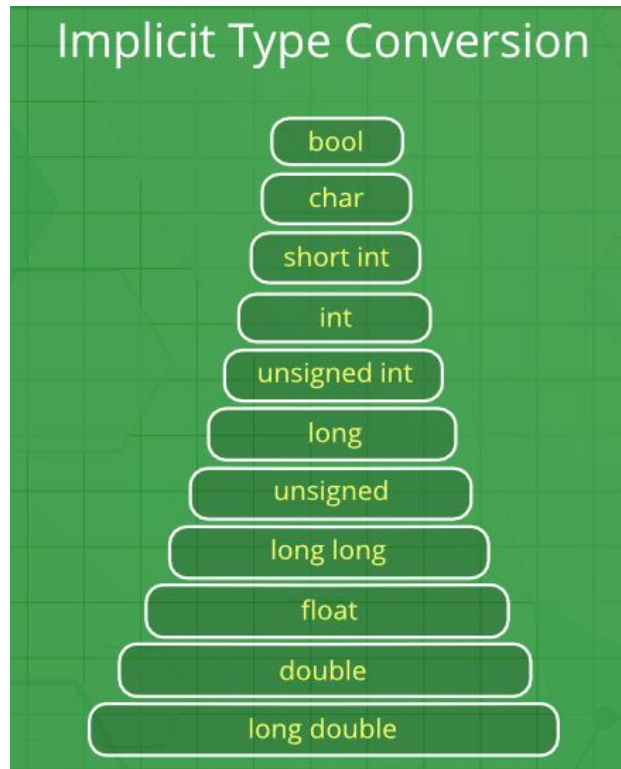
`40 + 2.0`

Είναι ισοδύναμο με:

`(double)40 + 2.0`

Ισοδύναμο με:

`40.0 + 2.0`



Τελεστής Ανάθεσης (Assignment Operator)

Ο **τελεστής ανάθεσης** παίρνει την τιμή του δεξιού τελεστέου (**rvalue**), την αναθέτει στην **μεταβλητή** του αριστερού τελεστέου (**lvalue**) και την επιστρέφει

Παράδειγμα:

`x = 42` // αναθέτει 42 στην x και επιστρέφει 42

Μπορούμε να αναθέσουμε πολλές μεταβλητές ταυτόχρονα:

`x = y = 42`

Ή ισοδύναμα:

`x = (y = 42)`

Συνδυαστικοί Τελεστές Ανάθεσης

Οι **συνδυαστικοί τελεστές ανάθεσης** επιτρέπουν να κάνουμε αριθμητικές πράξεις και να σώσουμε το αποτέλεσμα με συντομογραφία. Γενική μορφή:

τελεστήος1 op= τελεστήος2

Όπου op είναι οποιοσδήποτε από τους τελεστές +, -, *, %, /, &, ^, |, <<, >>

Παράδειγμα:

a += b

Είναι ισοδύναμο με:

a = a + b

Αντίστοιχα, **a *= b** είναι ισοδύναμο με **a = a * b** και ομοίως για τους άλλους τελεστές

Τελεστές Αύξησης και Μείωσης

Ο **τελεστής αύξησης ++** μπαίνει πριν ή μετά από το όνομα μίας μεταβλητής και την αυξάνει κατά 1. Ο **τελεστής μείωσης --** μπαίνει πριν ή μετά από το όνομα μίας μεταβλητής και την μειώνει κατά 1.

Επιθεματικά:

a++ επιστρέφει την τιμή του **a** πριν την αύξηση

a-- επιστρέφει την τιμή του **a** πριν την μείωση

Προθεματικά:

++a επιστρέφει την τιμή του **a** μετά την αύξηση

--a επιστρέφει την τιμή του **a** μετά την μείωση

Τελεστής Παράθεσης (Comma Operator)

Ο **τελεστής παράθεσης** υπολογίζει τον πρώτο τελεστέο, στην συνέχεια αγνοεί το αποτέλεσμα και επιστρέφει τον δεύτερο τελεστέο.

Παράδειγμα:

12, 42 επιστρέφει 42

Σχετικά περιορισμένες χρήσεις συνήθως χρησιμοποιείται με τελεστές ανάθεσης.

Πρόγραμμα Υπολογισμού Βαθμολογίας

```
int grade(int final_exam, int homework, int lab){  
    return final_exam*50/100+homework*30/100+lab*20/100;  
}  
  
int main(int argc, char ** argv) {  
    if (argc != 4) {  
        printf("Run as: grade final_exam homework lab\n");  
        return 1;  
    }  
  
    int final_exam = atoi(argv[1]);  
    int homework = atoi(argv[2]);  
    int lab = atoi(argv[3]);  
  
    int final_grade = grade(final_exam, homework, lab);  
  
    printf("My grade is: %d\n", final_grade);  
  
    return 0;  
}
```

Προτεραιότητα (Precedence) & Προσεταιριστικότητα (Associativity)

Η **προτεραιότητα** ενός τελεστή καθορίζει την σειρά με την οποία θα εκτελεστούν οι υπολογισμοί. Για παράδειγμα, ο πολλαπλασιασμός έχει υψηλότερη προτεραιότητα από την πρόσθεση:

$5 * 6 + 3 * 4$ επιστρέφει ??

Αν δύο τελεστές έχουν την ίδια προτεραιότητα, η **προσεταιριστικότητα** τους καθορίζει την σειρά των υπολογισμών (**αριστερά προς τα δεξιά ή το αντίστροφο**). Για παράδειγμα, ο πολλαπλασιασμός και η διαίρεση έχουν την ίδια προτεραιότητα, ενώ η προσεταιριστικότητά τους είναι από αριστερά προς τα δεξιά

$90 * 50 / 100$ επιστρέφει ??

Προτεραιότητα (Precedence) & Προσεταιριστικότητα (Associativity)

Η **προτεραιότητα** ενός τελεστή καθορίζει την σειρά με την οποία θα εκτελεστούν οι υπολογισμοί. Για παράδειγμα, ο πολλαπλασιασμός έχει υψηλότερη προτεραιότητα από την πρόσθεση:

$5 * 6 + 3 * 4$ επιστρέφει ??

Αν δύο τελεστές έχουν την ίδια προτεραιότητα, η **προσεταιριστικότητα** τους καθορίζει την σειρά των υπολογισμών (**αριστερά προς τα δεξιά ή το αντίστροφο**). Για παράδειγμα, ο πολλαπλασιασμός και η διαίρεση έχουν την ίδια προτεραιότητα, ενώ η προσεταιριστικότητά τους είναι από αριστερά προς τα δεξιά

$90 * 50 / 100$ επιστρέφει ??

Δεν είμαστε σίγουροι για την ακριβή σειρά; Χρησιμοποιούμε παρενθέσεις (!)

Θέση	Τελεστές	Προσεταιριστικότητα
1	() (παρενθέσεις-κλήση συνάρτησης) [] -> . ++ (επιθεματική αύξηση) -- (επιθεματική μείωση)	αριστερά προς δεξιά
2	++ (προθεματική αύξηση) -- (προθεματική μείωση) ! ~ * (έμμεση αναφορά) & (διεύθυνση) + (μοναδιαίο +) - (μοναδιαίο -) (προσαρμογή τύπου) sizeof	δεξιά προς αριστερά
3	* (πολλαπλασιασμός) / %	αριστερά προς δεξιά
4	+ (πρόσθεση) - (αφαίρεση)	αριστερά προς δεξιά
5	<< >>	αριστερά προς δεξιά
6	< <= > >=	αριστερά προς δεξιά
7	== !=	αριστερά προς δεξιά
8	&	αριστερά προς δεξιά
9	^	αριστερά προς δεξιά
10		αριστερά προς δεξιά
11	&&	αριστερά προς δεξιά
12		αριστερά προς δεξιά
13	?:	δεξιά προς αριστερά
14	= += -= *= /= %= &= ^= = <<= >>=	δεξιά προς αριστερά
15	,	αριστερά προς δεξιά

Εκφράσεις και Εντολές (Expressions and Statements)

Κάθε έγκυρος συνδυασμός τελεστών και τελεστέων (μεταβλητών, σταθερών) λέγεται **έκφραση (expression)**. Παράδειγμα έκφρασης:

```
final_exam * 50 / 100 + homework * 30 / 100 + lab * 20 / 100;
```

Μια συνάρτηση αποτελείται από ένα εκφράσεις που συνδυάζονται και εκτελούνται με τον τρόπο που καθορίζουν συντακτικές δομές που λέγονται **εντολές (statements)**. Οι εντολές εκτελούνται διαδοχικά, υπό συνθήκη ή επανειλημμένα, Παράδειγμα:

```
if (argc != 4) {  
    ...  
}
```


Κατηγοριοποίηση Εντολών

1. Κενές Εντολές
2. Εντολές Έκφρασης
3. Σύνθετες Εντολές
4. Εντολές Συνθήκης
5. Εντολές Επανάληψης

Κενή Εντολή (Empty / Null Statement)

Η κενή εντολή δεν εκτελεί τίποτε (no-operation ή no-op) και έχει την μορφή:

```
;    // null statement
```

```
;;;;;    // 5 null statements
```

Η χρησιμότητά της φαίνεται σε συνδυασμό με άλλες εντολές.

Σημείωση: χρησιμοποιούμε semicolon (;) ως διαχωριστικό εντολών - για να δείξουμε που τελειώνει η εντολή μας.

Εντολή Έκφρασης (Expression Statement)

Εντολή έκφρασης στην C ονομάζουμε μια έκφραση που τελειώνει με semicolon (;).
Παραδείγματα expression statements:

```
x = 4;
```

```
y = 7;
```

```
z = ++y;
```

```
y = z - (x++);
```

```
z = x - (--y);
```

Ποια η τιμή των x, y, z μετά την εκτέλεση αυτών των εντολών έκφρασης;

Σύνθετη Εντολή (Compound Statement / Block)

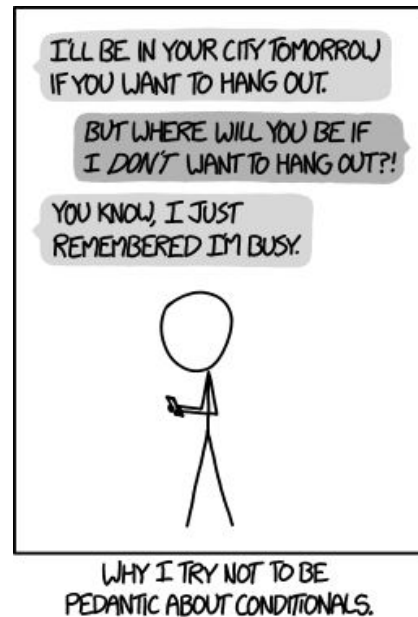
Σύνθεση εντολή ή αλλιώς **block**, λέγεται μια σειρά από εντολές όταν περικλείονται από αγκύλες { }.

Παράδειγμα:

```
{  
    x = 4;  
    y = 7;  
    z = ++y;  
    y = z - (x++);  
    z = x - (--y);  
}
```

Εντολές Ροής Ελέγχου (Control Flow Statements)

Η **ροή ελέγχου**, δηλαδή η σειρά με την οποία θα εκτελεστούν οι εντολές σε ένα πρόγραμμα καθορίζεται από τις εντολές ροής ελέγχου. Συνήθως οπτικοποιείται με ένα διάγραμμα.



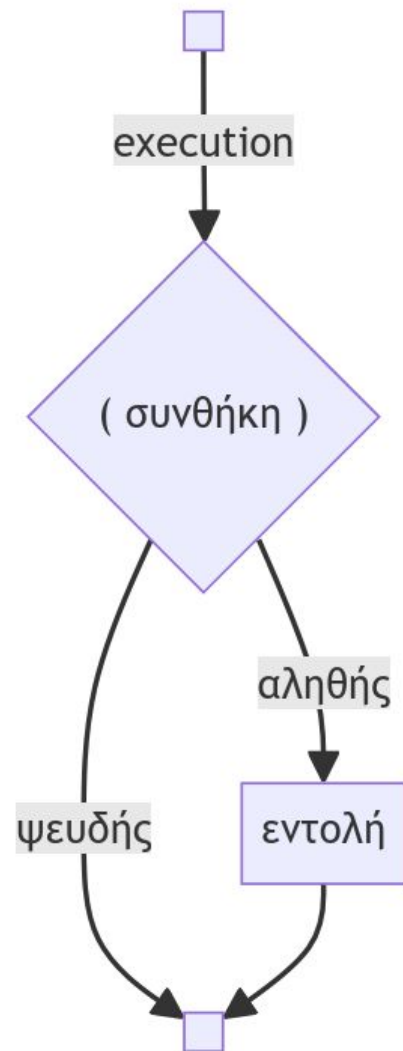
Εντολή if (if Statement)

Η εντολή if εκτελεί μια εντολή αν μια **λογική** συνθήκη είναι αληθής. Γενική μορφή:

if (συνθήκη)
εντολή

Παραδείγματα:

<pre>if (year > 1) ;</pre>	<pre>if (year > 1) year++;</pre>	<pre>if (year > 1) { year++; }</pre>
-----------------------------------	---	---



Εντολή if (if Statement)

Η εντολή if εκτελεί μια εντολή αν μια **λογική** συνθήκη είναι αληθής. Γενική μορφή:

if (συνθήκη)
εντολή

Οι παρενθέσεις είναι υποχρεωτικές!

Παραδείγματα:

```
if (year > 1)  
    ;
```

```
if (year > 1)  
    year++;
```

```
if (year > 1) {  
    year++;  
}
```



Εντολή if-else (if-else Statement)

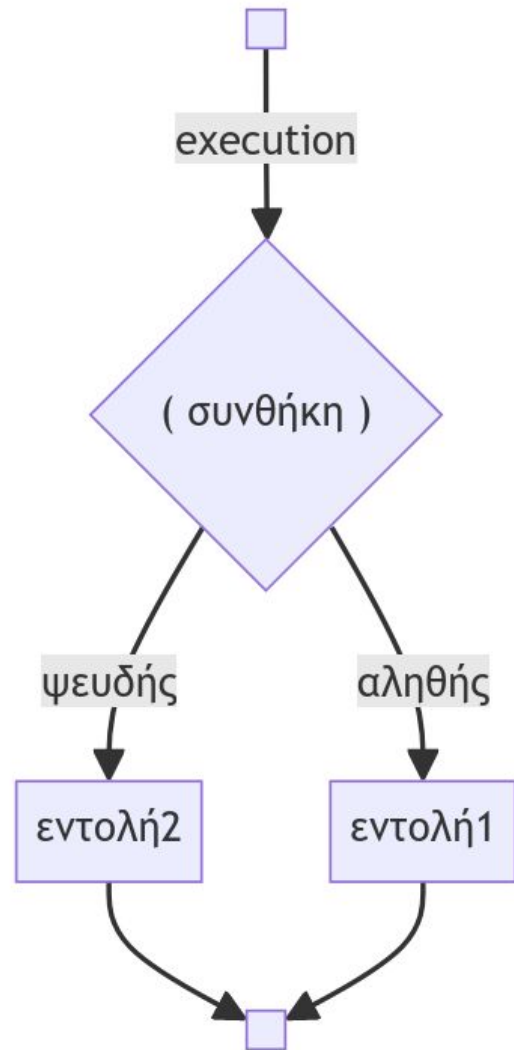
Επέκταση της εντολής if προκειμένου να εκτελέσουμε μια άλλη εντολή αν η λογική συνθήκη είναι ψευδής.

if (συνθήκη)

εντολή1

else

εντολή2



Εντολή if-else (if-else Statement)

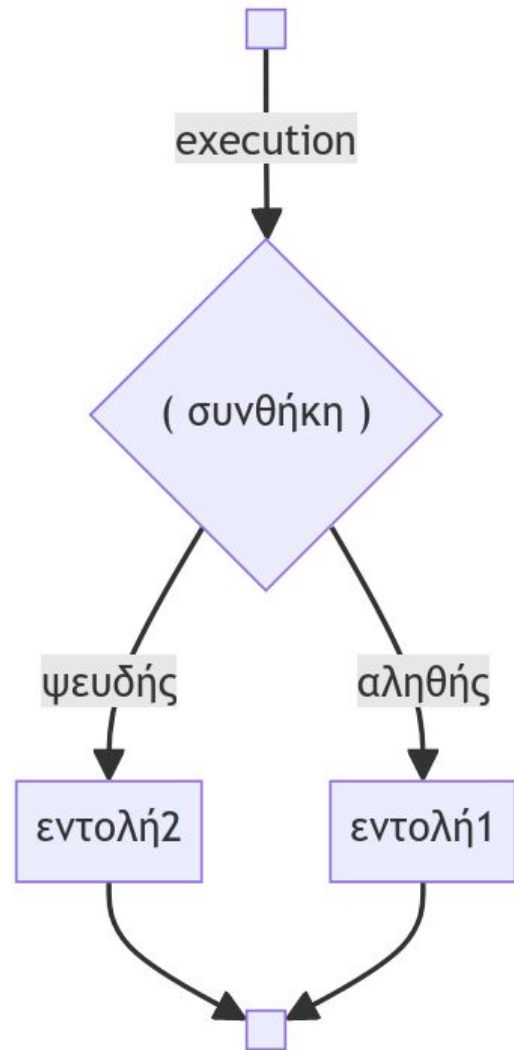
Παραδείγμα:

```
if (x > y)
    max = x;
else
    max = y;
```

Θα μπορούσαμε να "ζήσουμε" χωρίς else;

Υπάρχει εναλλακτικός τρόπος να γράψουμε το παραπάνω;

Τι κάνουμε όταν έχουμε πάνω από δύο συνθήκες;



Για την Επόμενη Φορά

- Από τις σημειώσεις του κ. Σταματόπουλου συνιστώ να έχετε καλύψει τα πάντα μέχρι την σελίδα 62.
- [Control flow](#)
- [Conditional statements](#) in programming languages

Ευχαριστώ και καλό Σαββατοκύριακο εύχομαι!
Keep (or Start!) coding ;)