

# Διάλεξη 4 - Τελεστές

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Εισαγωγή στον Προγραμματισμό

Θανάσης Αυγερινός

# Ανακοινώσεις / Διευκρινήσεις

- Tips & Tricks: Windows snipping tool

## Την Προηγούμενη Φορά

- Μνήμη στους υπολογιστές
- Δηλώσεις μεταβλητών
- Αναπαράσταση μεταβλητών στην μνήμη
- Τύπων μεταβλητών

# Διάλεξη 4 - Τελεστές

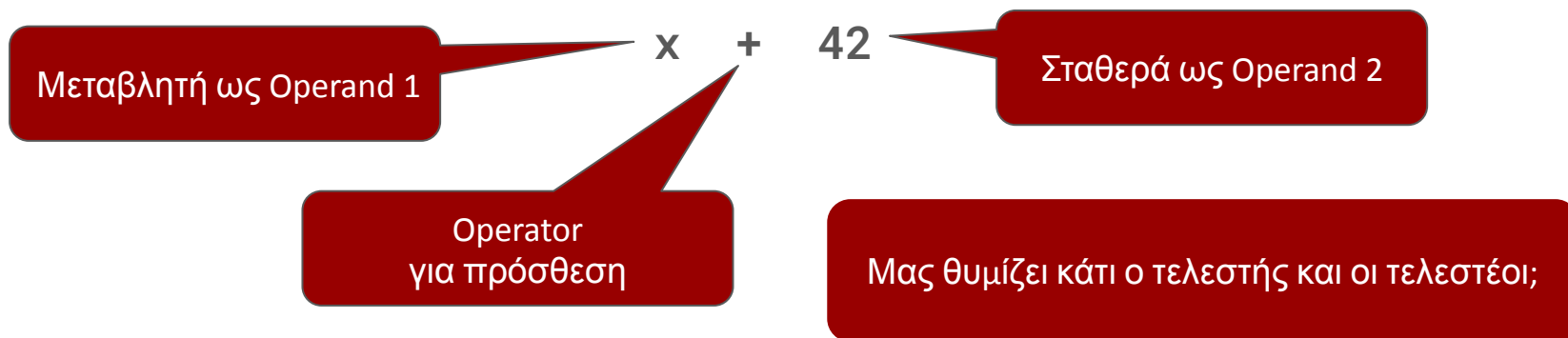
Σήμερα:

- Τελεστές
  - Μοναδιαίοι, Δυαδικοί, Τριαδικοί
  - Αριθμητικοί, Συγκριτικοί, Λογικοί, Συνθήκης, Bitwise, Μετατροπής, Ανάθεσης
  - Προτεραιότητα & Προσεταιριστικότητα

# Τελεστής (Operator) και Τελεστέος (Operand)

Οι **ΤΕΛΕΣΤΕΣ** στην C είναι σύμβολα που πραγματοποιούν μαθηματικούς, συγκριτικούς, δυαδικούς, υποθετικούς ή λογικούς υπολογισμούς.

Οι **ΤΕΛΕΣΤΕΟΙ** στην C είναι τα αντικείμενα (μεταβλητές, σταθερές) πάνω στα οποία πραγματοποιούν τους υπολογισμούς οι τελεστές.



# Πρόγραμμα - Συνάρτηση - Τελεστές

Το πρόγραμμα παίρνει κάποια δεδομένα εισόδου και υπολογίζει μια έξοδο

Το πρόγραμμα αποτελείται από συναρτήσεις που παίρνουν ορίσματα και υπολογίζουν μια έξοδο

Οι συναρτήσεις χρησιμοποιούν τελεστές και τελεστέους προκειμένου να υπολογίσουν μια έξοδο



*It's turtles all the way down!*

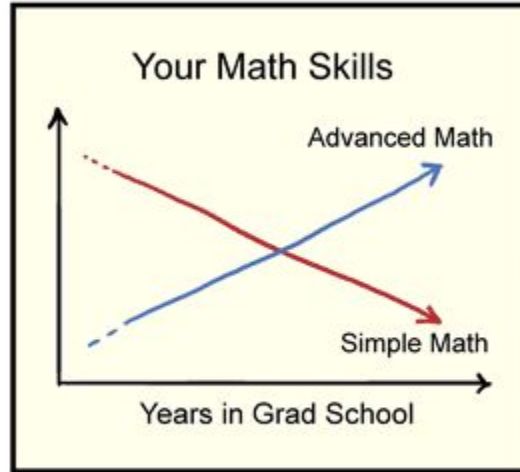
# Κατηγοριοποίηση Τελεστών #1

Κατηγοριοποίηση με βάση το [arity](#) - πόσους τελεστέους έχουν:

- Μοναδιαίοι Τελεστές (Unary Operators): Ένα τελεστή (operand)
- Δυαδικοί Τελεστές (Binary Operators): Δύο τελεστές
- Τριαδικοί Τελεστές (Ternary Operators): Τρεις τελεστές

Στην συνέχεια θα δούμε κατηγοριοποίηση με βάση την χρήση του τελεστή και στην σειρά υπολογισμού

# 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
<b>&amp;&amp;</b>	Λογική σύζευξη (AND, $\wedge$ ) είναι και οι δύο τελεστέοι αληθείς;	$42 > 0 \ \&\& \ 2 > 3$ επιστρέφει 0 $42 > 0 \ \&\& \ 3 > 2$ επιστρέφει 1
<b>  </b>	Λογική διάζευξη (OR, $\vee$ ) είναι ένας εκ των δύο τελεστέων αληθής;	$2 > 3 \    \ 42 > 0$ επιστρέφει 1 $2 > 3 \    \ 42 < 0$ επιστρέφει 0
<b>!</b>	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

# 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:

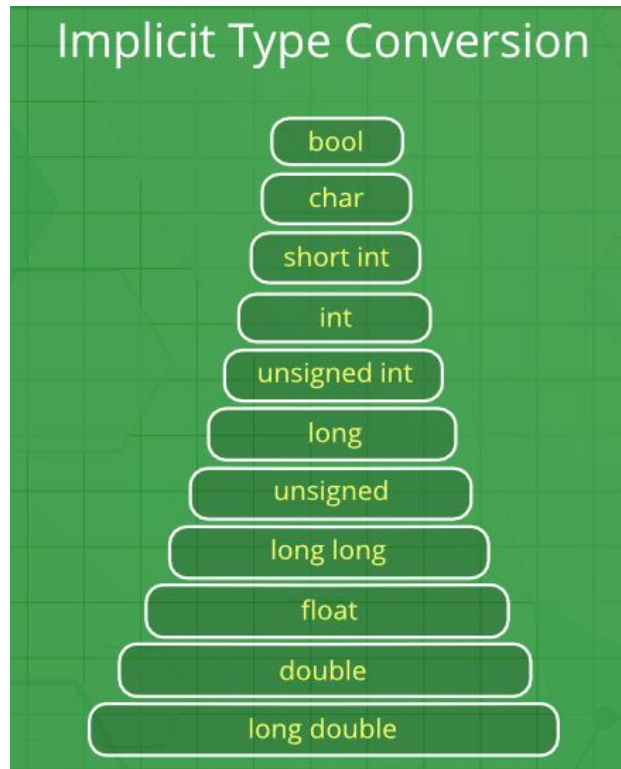
`40 + 2.0`

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

```
#include <stdio.h>
#include <stdlib.h>

// Compute grades using the class formula
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("Program needs to be called as `./prog final_exam homework lab`\n");
        return 1;
    }
    int final_exam = atoi(argv[1]);
    int homework = atoi(argv[2]);
    int lab = atoi(argv[3]);
    printf("Grade: %d\n", grade(final_exam, homework, lab));
    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 (!)



Προτεραιότητα αυξάνει

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) <b>sizeof</b>	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

Τι θα τυπώσει το παρακάτω πρόγραμμα;

```
#include <stdio.h>
int main() {
    float a = 3.1;

    if(a == 3.1)
        printf("OK\n");
    else
        printf("Not OK\n");
    return 0;
}
```

Υπάρχει κάποια εξήγηση;



## Για την Επόμενη Φορά

- Από τις σημειώσεις του κ. Σταματόπουλου συνιστώ να έχετε καλύψει τα πάντα μέχρι την σελίδα 43.
- [Type conversion](#)
- [List of operators](#)
- [lvalues and rvalues](#)
- [Precedence and associativity](#)

Ευχαριστώ και καλό σαβκο εύχομαι!  
Keep Coding ;)