

343 – Εισαγωγή στον Προγραμματισμό

Τμήμα Μαθηματικών
Πανεπιστήμιο Ιωαννίνων

Ακαδημαϊκό Έτος 2015-2016

Χάρης Παπαδόπουλος
207δ, Β' όροφος
e-mail: charis@cs.uoi.gr

Ωρες Γραφείου:
Πέμπτη 11-13

Διδασκαλία

- Κάθε εβδομάδα
 - 3ωρη διάλεξη: ανάλυση χαρακτηριστικών της C/C++
 - 2ωρο εργαστήριο: πρακτική εξάσκηση (υλοποίηση προγραμμάτων) μέσα από συγκεκριμένα προβλήματα
- Εργαστήρια:
 - 7 εβδομαδιαία εργαστήρια (5+2Quiz)
 - Σκοπός των εργαστηρίων είναι η εμπέδωση της ύλης, η εξάσκηση και η λύση αποριών
 - Η παρακολούθηση είναι υποχρεωτική και είναι το *διαβατήριο* για να λάβετε μέρος στις τελικές εξετάσεις
 - Μπορείτε να λείψετε σε **1 το πολύ** εργαστήριο
 - Προσοχή: Εάν λείψετε σε παραπάνω εργαστήρια δεν μπορείτε να λάβετε μέρος στις εξετάσεις !!!

Διδασκαλία

- Κάθε εβδομάδα
 - 3ωρη διάλεξη: ανάλυση χαρακτηριστικών της C/C++
 - 2ωρο εργαστήριο: πρακτική εξάσκηση (υλοποίηση προγραμμάτων) μέσα από συγκεκριμένα προβλήματα
- Εργαστήρια:
 - 8** – ~~7~~ εβδομαδιαία εργαστήρια ~~(5+2Quiz)~~ **6**
 - Σκοπός των εργαστηρίων είναι η εμπέδωση της ύλης, η εξάσκηση και η λύση αποριών
 - Η παρακολούθηση είναι υποχρεωτική και είναι το *διαβατήριο* για να λάβετε μέρος στις τελικές εξετάσεις
 - Μπορείτε να λείψετε σε **1 το πολύ** εργαστήριο
 - Προσοχή: Εάν λείψετε σε παραπάνω εργαστήρια δεν μπορείτε να λάβετε μέρος στις εξετάσεις !!!

Θ: διάλεξη (θεωρία)

Ε: Εργαστήριο

Q: Τεστ quiz

Ημερολόγιο Μαθήματος

Οκτώβριος 2015

Δ	Τ	Τ	Π	Π
			1	2
5	6	7	8	9 Θ
12	13	14	15	16 Θ
19 Ε	20	21	22	23 Θ
26	27	28	29	30 Θ

Νοέμβριος 2015

Δ	Τ	Τ	Π	Π
2 Ε	3	4	5	6 Θ
9 Ε	10	11	12	13 Θ
16 Q	17	18	19	20 Θ
23 Ε	24	25	26	27 Θ
30 Ε				

Δεκέμβριος 2015

Δ	Τ	Τ	Π	Π
	1	2	3	4 Θ
7 Ε	8	9	10	11 Θ
14 Q	15	16	17	18 Θ

Ιανουάριος 2016

Δ	Τ	Τ	Π	Π
4	5	6	7	8
11	12	13	14	15 Θ

Εβδομάδα	Θέματα	Υψη βιβλιογραφίας
Πα, 9 Οκτωβρίου	Εισαγωγικά μαθήματος & Δυσαική αναπαράσταση	[1]: 1.1, Παράρτημα 3 [2]: Κεφ. 1, Β, Δ
Πα, 16 Οκτωβρίου	Είσοδος/Εξοδος δεδομένων, τύποι δεδομένων & μεταβλητών	[1]: 1.2, 1.3, 1.4, 1.5, Παράρτημα 1 [2]: Κεφ. 2, Γ
Δε, 19 Οκτ	1 ^ο Εργαστήριο	
Πα, 23 Οκτωβρίου	Προεπεξεργαστής, αριθμητικοί και λογικοί τελεστές	[1]: 2.1, Παράρτημα 2 [2]: 4.11, 4.12, Α, ΣΤ
Πα, 30 Οκτωβρίου	Ροή ελέγχου: if/else, switch, for, while, do-while και ροή ελέγχου if/else	[1]: 2.2, 2.3 [2]: Κεφ. 4, Κεφ. 5
Δε, 2 Νοε	2 ^ο Εργαστήριο	
Πα, 6 Νοεμβρίου	Συναρτήσεις, εμβέλεια μεταβλητών και αναδρομή	[1]: 3.1, 3.2, 3.3, 4.1, 4.2, 13.1, 13.2 [2]: Κεφ. 6
Δε, 9 Νοε	3 ^ο Εργαστήριο	
Πα, 13 Νοεμβρίου	Επανάληψη Εργαστηρίων	
Δε, 16 Νοε	1 ^ο Quiz	
Πα, 20 Νοεμβρίου	Επανάληψη με Παραδείγματα	
Δε, 23 Νοε	4 ^ο Εργαστήριο	
Πα, 27 Νοεμβρίου	Πίνακες (μονοδιάστατοι και πολυδιάστατοι)	[1]: 5.1, 5.2, 5.4 [2]: Κεφ. 7
Δε, 30 Νοε	5 ^ο Εργαστήριο	
Πα, 4 Δεκεμβρίου	Εφαρμογές σε ταξινομήσεις και αναζήτηση στοιχείων	[1]: Παράρτημα 4, 9.1, 9.2, 9.3 [2]: 6.7, 6.8, Κεφ. 18
Δε, 7 Δεκ	6 ^ο Εργαστήριο	
Πα, 11 Δεκεμβρίου	Αλφαριθμητικά και Συμβολοσειρές	[1]: 6.1, 12.1, 12.2, 12.4 [2]: Κεφ. 21, 17.1-17.10
Δε, 14 Δεκ	2 ^ο Quiz	
Πα, 18 Δεκεμβρίου	Εγγραφές, δομές και χρήση αρχείων	[1]: 5.3, 13.3 [2]: 7.7, 7.8, 8.6, Κεφ. 19
Πα, 15 Ιανουαρίου	Επανάληψη	

Θ: διάλεξη (θεωρία)

Ε: Εργαστήριο

Q: Τεστ quiz

Ημερολόγιο Μαθήματος

Οκτώβριος 2015

Δ	Τ	Τ	Π	Π
			1	2
5	6	7	8	9 Θ
12	13	14	15	16 Θ
19 Ε	20	21	22	23 Θ
26	27	28	29	30 Θ

Νοέμβριος 2015

Δ	Τ	Τ	Π	Π
2 Ε	3	4	5	6 Θ
9 Ε	10	11	12	13 Θ
16 Q	17	18	19	20 Θ
23 Ε	24	25	26	27 Θ
30 Ε				

Δεκέμβριος 2015

Δ	Τ	Τ	Π	Π
	1	2	3	4 Θ
7 Ε	8	9	10	11 Θ
14 Q	15	16	17	18 Θ

Ιανουάριος 2016

Δ	Τ	Τ	Π	Π
4	5	6	7	8
11	12	13	14	15 Θ

Εβδομάδα	Θέματα	Υψη βιβλιογραφίας
Πα, 9 Οκτωβρίου	Εισαγωγικά μαθήματος & Δυσαική αναπαράσταση	[1]: 1.1, Παράρτημα 3 [2]: Κεφ. 1, Β, Δ
Πα, 16 Οκτωβρίου	Είσοδος/Εξοδος δεδομένων, τύποι δεδομένων & μεταβλητών	[1]: 1.2, 1.3, 1.4, 1.5, Παράρτημα 1 [2]: Κεφ. 2, Γ
Δε, 19 Οκτ	1 ^ο Εργαστήριο	
Πα, 23 Οκτωβρίου	Προεπεξεργαστής, αριθμητικοί και λογικοί τελεστές	[1]: 2.1, Παράρτημα 2 [2]: 4.11, 4.12, Α, ΣΤ
Πα, 30 Οκτωβρίου	Ροή ελέγχου: if/else, switch, for, while, do-while και ροή ελέγχου if/else	[1]: 2.2, 2.3 [2]: Κεφ. 4, Κεφ. 5
Δε, 2 Νοε	2 ^ο Εργαστήριο	
Πα, 6 Νοεμβρίου	Συναρτήσεις, εμβέλεια μεταβλητών και αναδρομή	[1]: 3.1, 3.2, 3.3, 4.1, 4.2, 13.1, 13.2 [2]: Κεφ. 6
Δε, 9 Νοε	3 ^ο Εργαστήριο	
Πα, 13 Νοεμβρίου	Επανάληψη Εργαστηρίων	
Δε, 16 Νοε	1 ^ο Quiz	
Πα, 20 Νοεμβρίου	Επανάληψη με Παραδείγματα	
Δε, 23 Νοε	4 ^ο Εργαστήριο	
Πα, 27 Νοεμβρίου	Πίνακες (μονοδιάστατοι και πολυδιάστατοι)	[1]: 5.1, 5.2, 5.4 [2]: Κεφ. 7
Δε, 30 Νοε	5 ^ο Εργαστήριο	
Πα, 4 Δεκεμβρίου	Εφαρμογές σε ταξινομήσεις και αναζήτηση στοιχείων	[1]: Παράρτημα 4, 9.1, 9.2, 9.3 [2]: 6.7, 6.8, Κεφ. 18
Δε, 7 Δεκ	6 ^ο Εργαστήριο	
Πα, 11 Δεκεμβρίου	Αλφαριθμητικά και Συμβολοσειρές	[1]: 6.1, 12.1, 12.2, 12.4 [2]: Κεφ. 21, 17.1-17.10
Δε, 14 Δεκ	2 ^ο Quiz	
Πα, 18 Δεκεμβρίου	Εγγραφές, δομές και χρήση αρχείων	[1]: 5.3, 13.3 [2]: 7.7, 7.8, 8.6, Κεφ. 19
Πα, 15 Ιανουαρίου	Επανάληψη	

Θ: διάλεξη (θεωρία)

Ε: Εργαστήριο

Q: Τεστ quiz

Ημερολόγιο Μαθήματος

Οκτώβριος 2015

Δ	Τ	Τ	Π	Π
			1	2
5	6	7	8	9 Θ
12	13	14	15	16 Θ
19 Ε	20	21	22	23 Θ
26	27	28	29	30 Θ

Νοέμβριος 2015

Δ	Τ	Τ	Π	Π
2 Ε	3	4	5	6 Θ
9 Ε	10	11	12	13 Θ
16 Q	17	18	19	20 Θ
23 Ε	24	25	26	27 Θ
30 Ε				

Δεκέμβριος 2015

Δ	Τ	Τ	Π	Π
	1	2	3	4 Θ
7 Ε	8	9	10	11 Θ
14 Q	15	16	17	18 Θ

Ιανουάριος 2016

Δ	Τ	Τ	Π	Π
4	5	6	7	8
11	12	13	14	15 Θ

Εβδομάδα	Θέματα	Υψη βιβλιογραφίας
Πα, 9 Οκτωβρίου	Εισαγωγικά μαθήματος & Δυσδική αναπαράσταση	[1]: 1.1, Παράρτημα 3 [2]: Κεφ. 1, Β, Δ
Πα, 16 Οκτωβρίου	Είσοδος/Εξοδος δεδομένων, τύποι δεδομένων & μεταβλητών	[1]: 1.2, 1.3, 1.4, 1.5, Παράρτημα 1 [2]: Κεφ. 2, Γ
Δε, 19 Οκτ	1 ^ο Εργαστήριο	
Πα, 23 Οκτωβρίου	Προεπεξεργαστής, αριθμητικοί και λογικοί τελεστές	[1]: 2.1, Παράρτημα 2 [2]: 4.11, 4.12, Α, ΣΤ
Πα, 30 Οκτωβρίου	Ροή ελέγχου: if/else, switch, for, while, do-while και ροή ελέγχου if/else	[1]: 2.2, 2.3 [2]: Κεφ. 4, Κεφ. 5
Δε, 2 Νοε	2 ^ο Εργαστήριο	
Πα, 6 Νοεμβρίου	Συναρτήσεις, εμβέλεια μεταβλητών και αναδρομή	[1]: 3.1, 3.2, 3.3, 4.1, 4.2, 13.1, 13.2 [2]: Κεφ. 6
Δε, 9 Νοε	3 ^ο Εργαστήριο	
Πα, 13 Νοεμβρίου	Επανάληψη Εργαστηρίων	
Δε, 16 Νοε	1 ^ο Quiz	
Πα, 20 Νοεμβρίου	Επανάληψη με Παραδείγματα	
Δε, 23 Νοε	4 ^ο Εργαστήριο	
Πα, 27 Νοεμβρίου	Πίνακες (μονοδιάστατοι και πολυδιάστατοι)	[1]: 5.1, 5.2, 5.4 [2]: Κεφ. 7
Δε, 30 Νοε	5 ^ο Εργαστήριο	
Πα, 4 Δεκεμβρίου	Εφαρμογές σε ταξινομήσεις και αναζήτηση στοιχείων	[1]: Παράρτημα 4, 9.1, 9.2, 9.3 [2]: 6.7, 6.8, Κεφ. 18
Δε, 7 Δεκ	6 ^ο Εργαστήριο	
Πα, 11 Δεκεμβρίου	Αλφαριθμητικά και Συμβολοσειρές	[1]: 6.1, 12.1, 12.2, 12.4 [2]: Κεφ. 21, 17.1-17.10
Δε, 14 Δεκ	2 ^ο Quiz	
Πα, 18 Δεκεμβρίου	Εγγραφές, δομές και χρήση αρχείων	[1]: 5.3, 13.3 [2]: 7.7, 7.8, 8.6, Κεφ. 19
Πα, 15 Ιανουαρίου	Επανάληψη	

Ενότητα 8

ΣΥΝΑΡΤΗΣΕΙΣ:

ΠΡΟΚΑΘΟΡΙΣΜΕΝΕΣ ΣΥΝΑΡΤΗΣΕΙΣ

Προκαθορισμένες Συναρτήσεις

- Δυο ειδών συναρτήσεις:
 - Αυτές που επιστρέφουν κάποια τιμή
 - Αυτές που δεν επιστρέφουν κάποια τιμή (`void`)
- Οι βιβλιοθήκες περιέχουν συναρτήσεις για να τις χρησιμοποιήσουμε
- Πρέπει να ανοίξουμε την κατάλληλη βιβλιοθήκη με `#include`,
π.χ.:
 - `<cmath>`, `<cstdlib>` (απλές "C" βιβλιοθήκες)
 - `<iostream>` (για `cout`, `cin`)

Προκαθορισμένες Συναρτήσεις

- Μαθηματικές συναρτήσεις:
 - Βρίσκονται στη βιβλιοθήκη `<cmath>`
 - Οι περισσότερες επιστρέφουν μια τιμή

```
theRoot = sqrt(9.0);
```

- Συστατικά:

`sqrt`: όνομα συνάρτησης

`theRoot`: μεταβλητή που αναθέτουμε την απάντηση
(απάντηση = επιστρεφόμενη τιμή)

`9.0`: όρισμα της συνάρτησης

`sqrt(9.0)`: κλήση της συνάρτησης

- **Είσοδος:** 9.0
- **Επεξεργασία:** υπολογισμός τετραγωνικής ρίζας
- **Έξοδος:** 3.0 που ανατίθεται στη μεταβλητή `theRoot`

Κλήση Συνάρτησης

Σύνταξη Συναρτήσεων που επιστρέφουν μια τιμή

όνομα_συνάρτησης(Λίστα_Ορισμάτων)

όπου Λίστα_Ορισμάτων:

όρισμα1, όρισμα2, ..., όρισμαN (N ≥ 0)

Παράδειγμα

```
side = sqrt(area);
```

```
cout << "2.5 εις την 3.0 = " << pow(2.5, 3.0) ;
```

- Η κλήση μπορεί να είναι μέρος μιας έκφρασης:

```
bonus = sqrt(sales)/10;
```

Παράδειγμα με συν/ση

```
#include <iostream>
#include <cmath>

int main( )
{
    const double COST_PER_SQ_MT = 10.50;
    double budget, area, lengthSide;

    cout << "Δώσε το διαθέσιμο ποσό ";
    cin >> budget;

    area = budget/COST_PER_SQ_MT;
    lengthSide = sqrt(area);

    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(2);
    cout << "Για την τιμή των " << budget << endl
    << " μπορώ να φτιάξω σκυλόσπιτο με << lengthSide
    << " μέτρα σε κάθε πλευρά.\n";
    return 0;
}
```

Παράδειγμα

Δώσε το διαθέσιμο ποσό **25**
Για την τιμή των 25.00 μπορώ
να φτιάξω σκυλόσπιτο με 1.54
μέτρα σε κάθε πλευρά.

Περισσότερες Προκαθορισμένες Συναρτήσεις

- `#include <cstdlib>`

- Περιέχει συναρτήσεις όπως:


- `abs()` // απόλυτη τιμή ενός `int`
- `labs()` // απόλυτη τιμή ενός `long int`
- `rand()` // τυχαίος ακέραιος

- `#include <cmath>`

- Περιέχει συναρτήσεις όπως:

- `pow(,)` // δύναμη a^b
- `floor()` // στρογγυλοποίηση προς τα κάτω
- `sqrt()` // τετραγωνική ρίζα

δέχεται δύο
ορίσματα



Πίνακας Μαθηματικών Συναρτήσεων

NAME	DESCRIPTION	TYPE OF ARGUMENTS	TYPE OF VALUE RETURNED	EXAMPLE	VALUE	LIBRARY HEADER
sqrt	Square root	double	double	sqrt(4.0)	2.0	cmath
pow	Powers	double	double	pow(2.0, 3.0)	8.0	cmath
abs	Absolute value for int	int	int	abs(-7) abs(7)	7 7	cstdlib
labs	Absolute value for long	long	long	labs(-70000) labs(70000)	70000 70000	cstdlib
fabs	Absolute value for double	double	double	fabs(-7.5) fabs(7.5)	7.5 7.5	cmath
ceil	Ceiling (round up)	double	double	ceil(3.2) ceil(3.9)	4.0 4.0	cmath
floor	Floor (round down)	double	double	floor(3.2) floor(3.9)	3.0 3.0	cmath
exit	End program	int	void	exit(1);	None	cstdlib
rand	Random number	None	int	rand()	Varies	cstdlib
srand	Set seed for rand	unsigned int	void	srand(42);	None	cstdlib

Υπάρχουν ΠΟΛΛΕΣ άλλες συναρτήσεις (τριγωνομετρικές, αλφαριθμητικές, ...)

ΠΑΡΑΡΤΗΜΑ 4

Προκαθορισμένες συναρτήσεις void

- Δεν επιστρέφουν τιμή
- Εκτελούν ορισμένες ενέργειες αλλά δεν στέλνουν κάποια απάντηση
- Όταν τις καλούμε αποτελούν μια εντολή:
`exit(1); // τερματίζει το πρόγραμμα`
- **Δήλωση:** `void exit(int n)`
 - έχει ένα όρισμα τύπου `int`
 - Κατά σύμβαση:
 - με όρισμα 1 υπάρχει σφάλμα κατά την εκτέλεση
 - με όρισμα 0 όλες οι υπόλοιπες περιπτώσεις
- **ΓΕΝΙΚΑ:** προσοχή στις μετατροπές τύπου στα ορίσματα

Γεννήτριες τυχαίων αριθμών

- Επιστρέφουν "τυχαία επιλεγμένους" αριθμούς
 - `rand()`
 - δεν δέχεται ορίσματα
 - Επιστρέφει ακέραιο μεταξύ 0 & `RAND_MAX`
 - `RAND_MAX`: σταθερά βιβλιοθήκης `<cstdlib>` εξαρτάται από το σύστημα, ~32767
 - Κλιμάκωση
 - Επιτρέπει επιλογή τυχαίων από μικρότερο διάστημα
`rand() % 6`
 - Επιστρέφει τυχαία τιμή μεταξύ 0 & 5
 - Μετατόπιση
`rand() % 6 + 1`
 - Μετατοπίζει το φάσμα μεταξύ 1 & 6 (π.χ., ρίψη ζαριού)

Γεννήτρια Συνάρτηση (seed)

- Ψευδοτυχαίοι αριθμοί
 - Πολλές κλήσεις της `rand()` \Rightarrow παράγουν την ίδια ακολουθία αριθμών
- Χρήση της γεννήτριας συνάρτησης (`seed`) για διαφορετικές τιμές:

```
srand(seed_value);
```

- `void` συνάρτηση
- Δέχεται ένα όρισμα, ακέραιο αριθμό "`seed`"
- Οποιαδήποτε `seed` τιμή, ακόμα και τον χρόνο συστήματος:

```
srand( time(0) );
```
- `time()` επιστρέφει τον χρόνο συστήματος σαν αριθμητική τιμή
- Βιβλιοθήκη `<time>` περιέχει συναρτήσεις `time()`

Παραδείγματα

- Τυχαίος double στο [0.0 ... 1.0]
`(RAND_MAX - rand())/static_cast<double>(RAND_MAX)`
 - Μετατροπή τύπων για να πάρουμε double αριθμό
- Τυχαίος ακέραιος μεταξύ 1 & 6:
`rand() % 6 + 1`
 - "%" υπόλοιπο διαίρεσης
- Τυχαίος ακέραιος μεταξύ 10 & 20:
`rand() % 10 + 10`

Παραδείγματα

- 10 όμοιες ακολουθίες τυχαίων ακεραίων

```
cout << "1st:\n";  
srand(99);  
for(i=0; i< 10; i++)  
    cout << (rand() % 11) << endl;  
  
cout << "2nd:\n";  
srand(99);  
for(i=0; i< 10; i++)  
    cout << (rand() % 11) << endl;
```

Παράδειγμα

```
1st:  
9  
3  
2  
10  
0  
1  
4  
9  
9  
7  
2nd:  
9  
3  
2  
10  
0  
1  
4  
9  
9  
7
```

```

char ans;
cin >> month >> day;
srand(month*day);
cout << "Weather for today:\n";
do
{
    prediction = rand() % 3;
    switch (prediction)
    {
        case 0:
            cout << "sunny! \n";
            break;
        case 1:
            cout << "cloudy!\n";
            break;
        case 2:
            cout << "stormy!\n";
            break;
        default:
            cout << "Program is not working.\n";
    }
    cout << "Next day?(y/n): ";
    cin >> ans;
}while (ans == 'y' || ans == 'Y');

```

Παράδειγμα

```

Give month day:
2 14
sunny!
Next day?(y/n): y
sunny!
Next day?(y/n): y
stormy!
Next day?(y/n): y
cloudy!
Next day?(y/n): n

```

Ενότητα 9

ΣΥΝΑΡΤΗΣΕΙΣ:

**ΟΡΙΖΟΜΕΝΕΣ ΑΠΟ ΤΟΝ
ΠΡΟΓΡΑΜΜΑΤΙΣΤΗ**

Συναρτήσεις οριζόμενες από εσάς

- Γράφετε τις δικές σας συναρτήσεις
- Χωρίζετε το πρόγραμμα σε κομμάτια
 - Δομημένος Προγραμματισμός
 - Επαναχρησιμοποίηση συναρτήσεων
 - Πιο εύκολο στην κατανόηση
- Ο ορισμός των συναρτήσεων μπορεί να γίνει:
 - είτε στο ίδιο αρχείο
 - είτε σε διαφορετικό αρχείο για να τις χρησιμοποιήσουν και άλλοι

Συστατικά για τον ορισμό συναρτήσεων

- 3 βασικά συστατικά
 - 1 **Δήλωση Συνάρτησης** (ή **πρωτότυπο** συν/σης)
Πληροφορία για τον μεταφραστή
 - 2 **Ορισμός Συνάρτησης**
Ο κώδικας της συνάρτησης για το τί ακριβώς κάνει
 - 3 **Κάλεσμα Συνάρτησης**
Μεταφέρει τον έλεγχο στην συνάρτηση

Δήλωση (πρωτότυπο) συνάρτησης

- Δήλωση για τον μεταφραστή
 - Καταλαβαίνει ότι πρόκειται για συνάρτηση
- Τοποθετείται πριν το πρώτο κάλεσμα της συνάρτησης
 - Συνήθως (σχεδόν πάντα) πριν τον ορισμό της `int main()`

Σύνταξη Δήλωσης Συνάρτησης

επιστρεφόμενος_τύπος όνομα_συνάρτησης(Λίστα_Ορισμάτων);

όπου Λίστα_Ορισμάτων:

όρισμα1, όρισμα2, ..., όρισμαN (N ≥ 0)

Παράδειγμα

```
double totalWeight( int number, double weight );
```

```
// Επιστρέφει το συνολικό βάρος τεμαχίων
```

```
// που το καθένα ζυγίζει weight
```

```
void showResults( double fdegrees, double cdegrees );
```

```
// Εκτυπώνει μήνυμα που λέει ότι οι fdegrees
```

```
// Φαρενάιτ ισοδυναμούν με cdegrees Κέλσιους
```

Ορισμός Συνάρτησης

- Κώδικας της συνάρτησης για το τί θα εκτελεί
 - όπως ακριβώς κάναμε μέσα στη `main()`
- Παράδειγμα
 - υπολογισμός συνολικού κόστους ($\#αντικειμένων * ποσό$) + 5% φόρος

Κεφαλίδα
συν/σης

```
double totalCost( int numberParameter, double priceParameter )  
{  
    const double TAXRATE = 0.05;  
    double subTotal;  
  
    subtotal = priceParameter * numberParameter;  
  
    return (subtotal + subtotal * TAXRATE);  
}
```

Σώμα
συν/σης

- Εντολή `return`:
 - επιστρέφει την έκφραση και τερματίζει την εκτέλεση της συν/σης

Τοποθέτηση Ορισμού Συνάρτησης

- Μετά την συνάρτηση `main()`
 - ΟΧΙ "μέσα" στη `main()`!
- Οι συναρτήσεις είναι "ισοδύναμες"
- Καμία συν/ση δεν αποτελεί κομμάτι κάποιας άλλης συν/σης
- Η εντολή `return`
 - Επιστρέφει δεδομένα σε αυτόν που κάλεσε την συν/ση

Κλήση Συνάρτησης

- Όπως και στις προκαθορισμένες συναρτήσεις:

```
bill = totalCost( number, price );
```

- Η συν/ση `totalCost()` επιστρέφει `double`
 - Ανατίθεται στη μεταβλητή `bill`
- Ορίσματα είναι: `number, price`
 - Μπορούν να είναι εκφράσεις, μεταβλητές, σταθερές ή και συνδυασμός αυτών
 - Στο κάλεσμα συν/σης, ένα όρισμα ονομάζεται και "τυπική παράμετρος"
- Μην συγχέεται τα ορίσματα των συναρτήσεων:
 - Ορίσματα σε λάθος σειρά οδηγούν σε λάθος αποτέλεσμα

```
#include <iostream>
using namespace std;
```

```
double totalCost(int numberParameter, double priceParameter);
//υπολογισμός συνολικού κόστους (#αντικειμένων*ποσό) + 5% φόρος
```

```
int main( )
{
    double price, bill;
    int number;

    cout << "Enter the number of items: ";
    cin >> number;
    cout << "Enter the price";
    cin >> price;

    bill = totalCost(number, price);

    cout << "Final bill: " << bill << endl;
    return 0;
}
```

```
double totalCost(int numberParameter, double priceParameter)
{
    const double TAXRATE = 0.05;
    double subtotal;
    subtotal = priceParameter * numberParameter;
    return (subtotal + subtotal*TAXRATE);
}
```

Δήλωση
συν/σης
(μην ξεχνάτε το ;
στο τέλος)

Κλήση
συν/σης

Κεφαλίδα
συν/σης

Σώμα
συν/σης

Ορισμός
συν/σης

Κλήση συναρτήσεων από συναρτήσεις

- Ήδη το κάνουμε αυτό:
 - η `main()` είναι συνάρτηση
- Η μοναδική απαίτηση:
 - Η δήλωση της συνάρτησης πρέπει να εμφανίζεται πρώτα
- Ο ορισμός της συνάρτησης πρέπει να εμφανίζεται
 - Μετά τον ορισμό της `main()`
- Συχνά συναρτήσεις καλούν άλλες συναρτήσεις
- Μια συνάρτηση μπορεί επίσης να καλεί τον εαυτό της → "Αναδρομή"

Παράδειγμα στρογγυλοποίησης

- Θέλουμε να στρογγυλοποιήσουμε έναν αριθμό στο πλησιέστερο ακέραιο τμήμα $2.4 \rightarrow 2$, $2.9 \rightarrow 3$
- Οι συν/σεις `floor()` και `ceil()` επιστρέφουν τον αμέσως προηγούμενο ή επόμενο ακέραιο αριθμό
 - `ceil(2.4) → 3` `floor(2.9) → 2`
- Αν προσθέσουμε 0.5 στον αριθμό και ζητήσουμε το `floor()`:
 - `floor(2.4+0.5) → 2` `floor(2.9+0.5) → 3`

```
int round(double number)
{
    return static_cast<int>(floor(number + 0.5));
}
```

```

#include <iostream>
#include <cmath>

int round(double number);

int main( )
{
    double doubleValue;
    char ans;

    do
    {
        cout << "Enter a double value: ";
        cin >> doubleValue;
        cout << "Rounded is " << round(doubleValue) << endl;
        cout << "Again? (y/n): ";
        cin >> ans;
    }while (ans == 'y' || ans == 'Y');

    return 0;
}

int round(double number)
{
    return static_cast<int>(floor(number + 0.5));
}

```

Τι εκτυπώνει το πρόγραμμα;

```
#include <iostream>

char mystery(int a, int b);

int main( )
{
    cout << mystery( 10,9 ) << "ow\n";
    return 0;
}

char mystery(int a, int b)
{
    if( a > b )
        return 'W';
    else
        return 'H';
}
```

- Γράψτε μια δήλωση και έναν ορισμό συν/σης που παίρνει τρία ορίσματα, όλα int, και επιστρέφει το άθροισμά τους.
- Γράψτε μια δήλωση και έναν ορισμό συν/σης που παίρνει ένα όρισμα, τύπου double. Η συν/ση επιστρέφει τον χαρακτήρα 'Θ' αν το όρισμα είναι θετικό και 'Α' αν το όρισμα είναι αρνητικό.

Συναρτήσεις που επιστρέφουν μια λογική τιμή

- Αντί να έχουμε

```
if ( (( rate >= 10 ) && ( rate < 20 )) || ( rate == 0 ) )  
{  
    ...  
}
```

- Θα μπορούσαμε

```
if ( appropriate(rate) )  
{  
    ...  
}
```

- Αρκεί να ορίσουμε και να δηλώσουμε την συν/ση:

```
bool appropriate(int rate)  
{  
    return ( (( rate >= 10 ) && ( rate < 20 )) || ( rate == 0 ) );  
}
```


Ορισμός συναρτήσεων τύπου void

- Συν/σεις τύπου void δεν επιστρέφουν τιμή
- 2 βασικές διαφορές:

λέξη void

&

δεν απαιτεί την εντολή return

```
void showResults ( double fdegrees, double cdegrees )
{
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(1);
    cout << fdegrees << " βαθμοί φαρενάιτ είναι ισοδύναμοι με "
        << cdegrees << " βαθμούς κέλσιους \n";
}
```

```
showResults ( 2.5, 0.3 ) ;
```

```

#include <iostream>

void iceCreamDivision(int number, double totalWeight);

int main( )
{
    int number;  double totalWeight;

    cout << "Enter the customers: ";  cin >> number;
    cout << "Enter weight of ice cream : ";  cin >> totalWeight;

    iceCreamDivision(number, totalWeight);

    return 0;
}

void iceCreamDivision(int number, double totalWeight)
{
    double portion;
    if (number == 0)
    {
        cout << "Cannot divide among zero customers.\n";
        return;
    }
    portion = totalWeight/number;
    cout << "Each one receives " << portion;
}

```

Μπορείτε να έχετε εντολή return μέσα σε συν/ση void.

Παραδείγματα

```
int main( )
{
    friendly( );
    shy( 6 );
    cout << " Ξανά: \n";
    shy( 2 );
    friendly( );
    cout << "Τέλος \n";
    return 0;
}

void friendly( )
{
    cout << "Γεια σας \n";
}

void shy( int a )
{
    if (a < 5 )
        return ;
    cout << "Αντίο \n";
}
```

Ενότητα 10

ΣΥΝΑΡΤΗΣΕΙΣ: ΚΑΝΟΝΕΣ ΕΜΒΕΛΕΙΑΣ

Τοπικές μεταβλητές

- **Τοπικές Μεταβλητές:** δηλώνονται μέσα σε μια συνάρτηση
- **Εμβέλεια:** μέσα σε αυτή την συνάρτηση

```
double totalCost( int numberParameter, double priceParameter )  
{  
    const double TAXRATE = 0.05;    _____ τοπική σταθερά  
    double subTotal;                _____ τοπική μεταβλητή  
  
    subtotal = priceParameter * numberParameter;  
  
    return (subtotal + subtotal * TAXRATE);  
}
```

- Αν μια μεταβλητή είναι τοπική τότε μπορείτε να έχετε μια άλλη μεταβλητή με το ίδιο όνομα σε μια άλλη συνάρτηση
 - δυο διαφορετικές μεταβλητές με το ίδιο όνομα

```

#include <iostream>

double estimate(int minPeas, int maxPeas, int podCount);

int main( ) // Επιστρέφει μια εκτίμηση
{ // για τη μέση σοδειά μπιζελιών

    int maxCount, minCount, podCount;
    double averagePea, yield;
    cout << "Enter minimum and maximum and number of pods: ";
    cin >> minCount >> maxCount >> podCount; // ελάχιστος-μέγιστος αριθμός
    cout << "Enter the average weight: "; // μπιζελιών σε έναν καρπό
    cin >> averagePea;
    yield = estimate(minCount, maxCount, podCount) * averagePea;
    cout << "Average weight = " << averagePea << endl
         << "Estimated average = " << yield << endl;
    return 0;
}

double estimate(int minPeas, int maxPeas, int podCount)
{
    double averagePea;

    averagePea = (maxPeas + minPeas)/2.0;
    return (podCount * averagePea);
}

```

Καθολικές μεταβλητές

- Οι τοπικές μεταβλητές έχουν εμβέλεια μόνο μέσα στη συνάρτηση που ορίζονται
- Μεταβλητές που δεν ορίζονται μέσα σε μια συνάρτηση ονομάζονται **καθολικές**
- Δηλώνονται στην αρχή του προγράμματος:
 - έξω από το σώμα όλων των συναρτήσεων
 - μπορούν να χρησιμοποιηθούν σε οποιαδήποτε συν/ση
- **Συνήθης τακτική:**
Δηλώνονται μετά τις οδηγίες `include < ... >` και την οδηγία `using ...`

```

#include <iostream>
#include <cmath>

const double PI = 3.14159; // καθολική μεταβλητή
double area(double radius); // εμβαδόν κύκλου
double volume(double radius); // όγκος σφαίρας

int main( )
{
    double radius, area, volume;
    cout << "Enter a radius \n"; cin >> radiusOfBoth;
    area = area(radius);
    volume = volume(radius);
    cout << "Area = " << area << "\nVolume = " << volume << "\n";
    return 0;
}

double area(double radius)
{
    return (PI * pow(radius, 2));
}

double volume(double radius)
{
    return ((4.0/3.0) * PI * pow(radius, 3));
}

```


Μπλοκ

- Σύνθετη εντολή (μπλοκ): τμήμα εντολών μέσα σε { }
- Μεταβλητή που δηλώνεται μέσα σε μπλοκ έχει εμβέλεια μόνο σε αυτό το μπλοκ
 - τοπική μεταβλητή του μπλοκ
- Ο ορισμός της μεταβλητής ισχύει από την δήλωση της μεταβλητής μέχρι την }
- Οι συναρτήσεις είναι μπλοκ

```
for(int i = 0; i < 10; i++)  
{  
    cout << i ;  
    cout << " , ";  
}
```

```
while(1)  
{  
    cout << "give a number:" ;  
    int num;  
    cin >> num;  
    if( num == 0 )  
        break;  
    else  
        cout << "non-zero!! \n";  
}
```

Κανόνες εμφάνισης

- Εμφωλιασμένα μπλοκ (και συναρτήσεις)
- Οι κανόνες ισχύουν και για καθολικές και τοπικές μεταβλητές

Κανόνας εμφάνισης

Αν ένα όνομα μεταβλητής χρησιμοποιείται σε δυο μπλοκ (το ένα μέσα στο άλλο) τότε είναι **δυο διαφορετικές μεταβλητές** με το ίδιο όνομα

- Η εσωτερική μεταβλητή δεν μπορεί να προσπελαστεί από το έξω μπλοκ
- Η εξωτερική μεταβλητή δεν μπορεί να προσπελαστεί από το μέσα μπλοκ
- Οποιαδήποτε αλλαγή δεν επηρεάζει την άλλη μεταβλητή

Παράδειγμα

```
{
    int x = 0;
    cout << x;
    {
        int x = 1;
        cout << x;
    }
}
```

Παράδειγμα εμφάνισης

```
{
  int x = 1;
  cout << x << endl;
  {
    cout << x;
    int x = 2;
    cout << x << endl;
    {
      cout << x;
      int x = 3;
      cout << x << endl;
    }
    cout << x << endl;
  }
  cout << x << endl;
}
```

Παράδειγμα εμφάνισης

```
#include <iostream>

int x = 1;
void funA( );
void funB( );

int main( )
{
    int x = 5;
    funA();
    funB();
    funA();
    funB();
    cout << x << " ";
}
```

```
void funA( )
{
    int x = 25;
    x++;
    cout << x << " ";
}

void funB( )
{
    x *= 10;
    cout << x << " ";
}
```

Ενότητα 11

ΣΥΝΑΡΤΗΣΕΙΣ:

ΑΝΑΔΡΟΜΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ

Αναδρομή

- Αναδρομικές συναρτήσεις
 - Συναρτήσεις που **καλούν το εαυτό τους**
 - Πρέπει να λύνουν την βασική περίπτωση χωρίς αναδρομή
- Διαιρούν το πρόβλημα σε
 - Βασική περίπτωση που λύνουν «εύκολα»
 - Σε άλλες περιπτώσεις που μοιάζουν με το αρχικό πρόβλημα και τότε
 - Καλούν ένα αντίγραφο του εαυτού τους για την λύση τους (**αναδρομή**)
- Κάποια στιγμή φτάνει στην βασική περίπτωση και την επιλύει
 - Περνάει επιμέρους αποτελέσματα προς τα επάνω και τελικά φτάνει στην λύση του όλου προβλήματος

Παράδειγμα αναδρομής: Παραγοντικό (n!)

- Παραγοντικό

- $5! = 5 * 4 * 3 * 2 * 1$

Παρατηρούμε

$$5! = 5 * 4!$$

$$4! = 4 * 3! \dots$$

Γενικά:

$$n! = n * (n-1)!$$

με $1! = 0! = 1$

- Μπορούν να υπολογιστούν αναδρομικά
 - Βάση αναδρομής ($1! = 0! = 1$). Μετά γυρίζουμε πίσω

$$2! = 2 * 1! = 2 * 1 = 2;$$

$$3! = 3 * 2! = 3 * 2 = 6;$$

$$4! = 4 * 3! = 4 * 6 = 24$$

Παραγοντικό

```
#include <iostream>
int factorial(int n);
int main( )
{
    for(int i = 1; i <= 20; i++)
        cout << "Factorial ( " << i << " ) = " << factorial(i);

    return 0;
}

int factorial(int n)
{
    if( n <= 1 )
        return 1;
    else
        return factorial(n-1) * n;
}
```


Η σειρά Fibonacci

- Η σειρά Fibonacci
 - 0, 1, 1, 2, 3, 5, 8...
 - Κάθε αριθμός είναι άθροισμα των δύο προηγούμενων

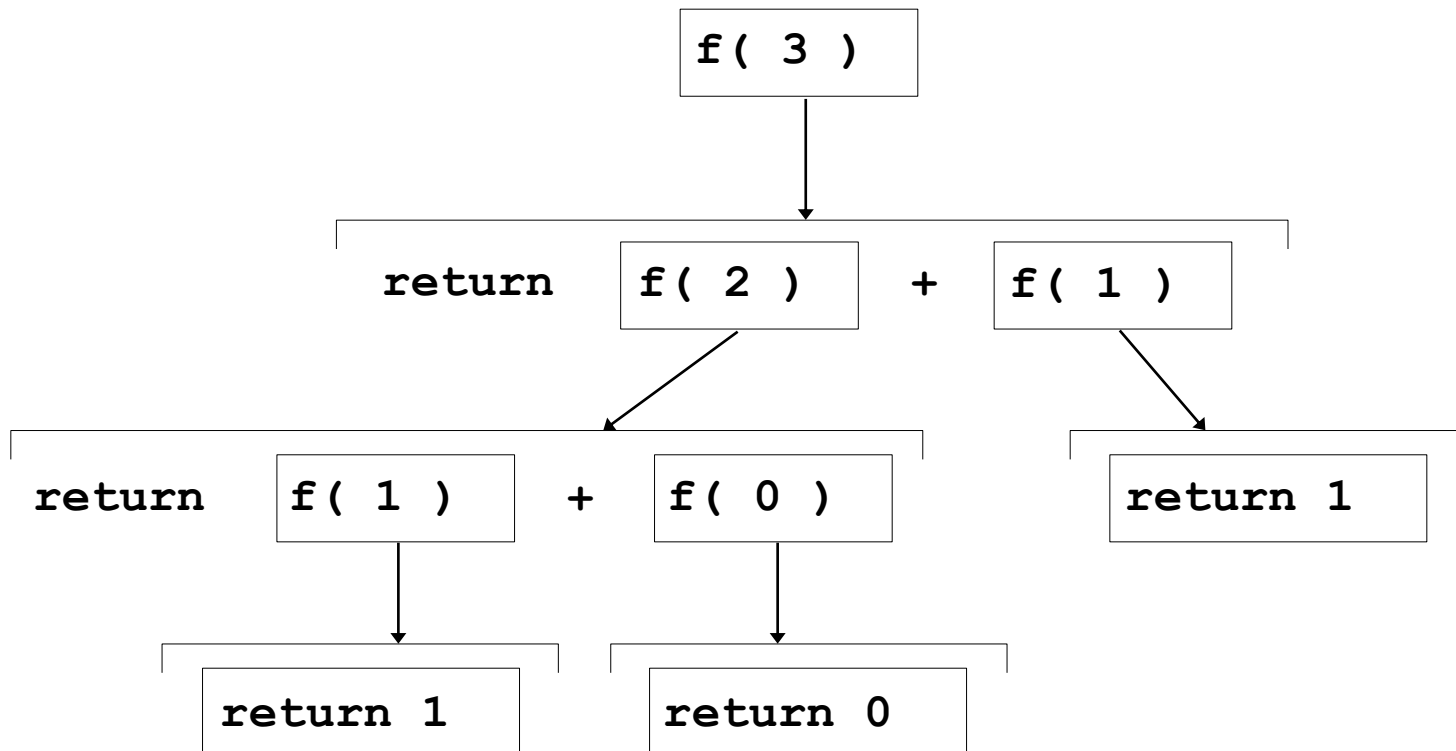
$$\text{fib}(n) = \text{fib}(n - 1) + \text{fib}(n - 2)$$

- Αναδρομικός ορισμός
 - Βάση: $\text{fib}(0) = 0$ και $\text{fib}(1) = 1$

Η σειρά Fibonacci

- Η σειρά Fibonacci
 - 0, 1, 1, 2, 3, 5, 8...
 - Κάθε αριθμός είναι άθροισμα των δύο προηγούμενων

$$\text{fib}(n) = \text{fib}(n - 1) + \text{fib}(n - 2)$$



Fibonacci

```
#include <iostream>
int fibonacci(int n);
int main( )
{
    cout << "Enter a number: ";
    cin >> num;

    cout << "Fibonacci ( " << num << " ) = " << fibonacci(num);

    return 0;
}

int fibonacci(int n)
{
    if( n <= 1 )
        return n;
    else
        return fibonacci( n - 1 ) + fibonacci( n - 2 );
}
```

Κάθετοι αριθμοί

- Στόχος: εμφάνισε τα ψηφία του αριθμού κάθετα, ένα σε κάθε γραμμή
- Παράδειγμα:
`writeVertical(1234);`

Επιθυμητό Αποτέλεσμα:

1
2
3
4

Αναδρομικός ορισμός

- Διάσπασε το πρόβλημα σε 2 περιπτώσεις
- Βάση: αν $n < 10$
 - Γράψε το αριθμό n στην οθόνη
- Αναδρομική περίπτωση: αν $n \geq 10$:
 - 1- Γράψε όλα τα ψηφία κάθετα εκτός από το τελευταίο
 - 2- Γράψε το τελευταίο ψηφίο
- Παράδειγμα: παράμετρος 1234:
 - 1^η ενέργεια: εμφανίζει τα 1, 2, 3 κάθετα
 - 2^η ενέργεια: εμφανίζει το 4

Ορισμός της writeVertical()

```
void writeVertical(int n)
{
    if (n < 10)                //Base case
        cout << n << endl;
    else
    {                            //Recursive step
        writeVertical(n/10);
        cout << (n%10) << endl;
    }
}
```

- Αναδρομική περίπτωση: αν $n \geq 10$:
 - 1- Γράψε όλα τα ψηφία κάθετα εκτός από το τελευταίο
 - 2- Γράψε το τελευταίο ψηφίο

Ανάλυση της writeVertical()

- Παράδειγμα εκτέλεσης:

```
writeVertical(123);  
→ writeVertical(12); (123/10)  
    → writeVertical(1); (12/10)  
        → cout << 1 << endl;  
    cout << 2 << endl;  
cout << 3 << endl;
```

- Τα → δείχνουν ποιον υπολογισμό εκτελεί η συν/ση
- Προσέξτε τις δυο πρώτες κλήσεις ξανά (αναδρομή)
- Η τελευταία κλήση (1) εκτυπώνει και "τερματίζει"

Μην ξεχνάτε την βάση της αναδρομής

- Θεωρήστε έναν άλλο τρόπο ορισμού της συν/σης:

```
void newWriteVertical(int n)
{
    newWriteVertical(n/10);
    cout << (n%10) << endl;
}
```

- Φαίνεται "λογική" συνάρτηση
- Του λείπει όμως η **βασική περίπτωση!!**
- Η αναδρομή δεν σταματάει ποτέ! **Ατέρμονη αναδρομή**

Θα μπορούσατε και χωρίς αναδρομή

```
void writeVertical(int n)
{
    int nsTens = 1;
    int leftEndPiece = n;
    while (leftEndPiece > 9)
    {
        leftEndPiece = leftEndPiece/10;
        nsTens = nsTens*10;
    }
    for (int i=nsTens; i > 0; i = i/10)
    {
        cout << (n/i) << endl;
        n = n%i;
    }
}
```

```
void writeVertical(int n)
{
    if (n < 10)
        cout << n << endl;
    else
    {
        writeVertical(n/10);
        cout << (n%10) << endl;
    }
}
```

Ενότητα 12

ΣΥΝΑΡΤΗΣΕΙΣ: ΠΑΡΑΜΕΤΡΟΙ

Παράμετροι κλήσης με τιμή

- Στην κλήση της συνάρτησης τοποθετούνται αντίγραφα των τιμών
- Ουσιαστικά είναι σαν τοπικές μεταβλητές
- Αν αλλάξουν τιμή, τότε "τοπικές" μεταβολές
 - Η συνάρτηση δεν έχει πρόσβαση στη "πραγματική παράμετρο"
- Αυτή είναι η εξ'ορισμού κλήση
 - σε όλα τα παραδείγματα ως τώρα

```

#include <iostream>
const double RATE = 150.00; //Ευρώ ανα 15 λεπτά.
double fee(int hoursWorked, int minutesWorked);
//Χρεώσεις για hoursWorked και minutesWorked

int main( )
{
    int hours, minutes;
    double bill;
    cout << "Enter the hours and minutes:\n";
    cin >> hours >> minutes;
    bill = fee(hours, minutes);
    cout << "For " << hours << " and " << minutes
         << " , your bill is " << bill << endl;
    return 0;
}

double fee(int hoursWorked, int minutesWorked)
{
    int quarterHours;

    minutesWorked = hoursWorked*60 + minutesWorked;
    quarterHours = minutesWorked/15;
    return (quarterHours*RATE);
}

```

Δεν αλλάζει η τιμή της μεταβλητής minutes από την κλήση της fee()

Τυπικό σφάλμα

- Κλασικό λάθος:

- Δήλωση παραμέτρου "ξανά" μέσα σε συν/ση:

```
double fee(int hoursWorked, int minutesWorked)
{
    int quarterHours;           // local variable
    int minutesWorked           // NO!
}
```

- Σφάλμα στον μεταφραστή:

- "Redefinition error..."

- **Παράμετροι με τιμή** είναι σαν "τοπικές μεταβλητές"

- Αλλά η συν/ση τις έχει δηλωμένες "αυτόματα"

Παράμετροι κλήσης με αναφορά

- Χρησιμοποιούνται για να έχουν πρόσβαση στις πραγματικές παραμέτρους
- Τα δεδομένα που καλεί κάποιος μπορεί να αλλάξουν κατά το κάλεσμα μιας συν/σης!
- Συνήθως χρησιμοποιούνται για είσοδο/διάβασμα
 - Αυτός που καλεί θέλει τις τιμές που θα διαβάσει η συν/ση
- Δηλώνονται με & μετά τον τύπο στη λίστα παραμέτρων

```
void getInput( double& receiver )
{
    cout << " Δώσε τιμή: \n";
    cin >> receiver;
}
```

```
int main()
{
    double number;
    ...
    getInput( number );
    ...
}
```

```

int main( )
{
    int firstNum, secondNum;
    getNumbers(firstNum, secondNum);
    swapValues(firstNum, secondNum);
    showResults(firstNum, secondNum);
}

void getNumbers(int& input1, int& input2)
{
    cout << "Enter two integers: ";
    cin >> input1 >> input2;
}

void swapValues(int& variable1, int& variable2)
{
    int temp;

    temp = variable1;
    variable1 = variable2;
    variable2 = temp;
}

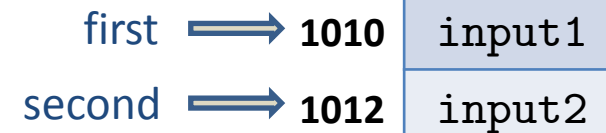
void showResults(int output1, int output2)
{
    cout << "Αντίστροφα:"<< output1 << " " << output2 << endl;
}

```

Μηχανισμός κλήσης με αναφορά

- Τι πραγματικά περνάμε σαν αναφορά;
- Μια αναφορά στην πραγματική παράμετρο όταν κάλεσε την συνάρτηση
 - Αναφέρεται στην θέση μνήμης της πραγματικής παραμέτρου
 - Καλείται συχνά ως "διεύθυνση", που είναι ένα μοναδικό νούμερο που δείχνει σε διακριτή θέση μνήμης

```
void getNumbers( int& input1, int& input2 );  
getNumbers( first, second)
```



Σταθερές Παράμετροι αναφοράς

- Οι παράμετροι με αναφορά εγκυμονούν κινδύνους
 - Τα δεδομένα μπορούν να αλλάξουν
 - Μερικές φορές είναι επιθυμητό, άλλες όχι
- Για να "προστατεύσουμε" τα δεδομένα, περνώντας παράλληλα παραμέτρους με αναφορά:
 - Χρήση της δεσμευμένης λέξης `const`

```
void sendConstRef( const int &par1,  
                  const int &par2 );
```

- Οι παράμετροι γίνονται "μόνο για διάβασμα" (read-only) από την συνάρτηση
- Δεν επιτρέπονται αλλαγές στο σώμα της συν/σης

Μικτές λίστες παραμέτρων

- Συνδυασμός παραμέτρων στο κάλεσμα της συν/σης
- Η λίστα παραμέτρων μπορεί να περιέχει παραμέτρους με τιμή και παραμέτρους με αναφορά
- Η σειρά των ορισμάτων στη λίστα είναι ΣΗΜΑΝΤΙΚΗ:

```
void mixedCall(int & par1, int par2, double & par3);
```

– Όταν καλούμε τη συν/ση:

```
mixedCall(arg1, arg2, arg3);
```

- arg1 integer, παράμετρος με αναφορά
- arg2 integer, παράμετρος με τιμή
- arg3 double, παράμετρος με αναφορά

```

#include <iostream>

void doStuff(int par1Value, int& par2Ref);

int main( )
{
    int n1, n2;

    n1 = 1;
    n2 = 2;

    doStuff(n1, n2);

    cout << "n1 μετά = " << n1 << endl;
    cout << "n2 μετά = " << n2 << endl;
    return 0;
}

void doStuff(int par1Value, int& par2Ref)
{
    par1Value = 111;
    cout << "par1Value στη συν/ση= " << par1Value << endl;
    par2Ref = 222;
    cout << "par2Ref στη συν/ση= " << par2Ref << endl;
}

```

Τι εκτυπώνει;

```
#include <iostream>

void figureMeOut(int& x, int y, int &z);

int main( )
{
    int a = 10, b = 20, c = 30;

    figureMeOut(a, b, c);

    cout << a << " " << b << " " << c << endl;
    return 0;
}

void figureMeOut(int& x, int y, int &z)
{
    cout << x << " " << y << " " << z << endl;
    x = 1;
    y = 2;
    z = 3;
    cout << x << " " << y << " " << z << endl;
}
```

Ενότητα 13

ΣΥΝΑΡΤΗΣΕΙΣ: ΥΠΕΡΦΟΡΤΩΣΗ

Υπερφόρτωση

- Συναρτήσεις με το ίδιο όνομα
- Διαφορετική λίστα παραμέτρων
- Δυο διαφορετικές δηλώσεις συναρτήσεων
- **Υπογραφή συνάρτησης**
 - Όνομα συνάρτησης & λίστα παραμέτρων
 - Μοναδικά για κάθε ορισμό συνάρτησης
- Επιτρέπει ίδια ενέργεια σε διαφορετικά δεδομένα

Παράδειγμα Υπερφόρτωσης: Μέσος Όρος

- Υπολογίζει το μέσο όρο 2 αριθμών:

```
double average(double n1, double n2)
{
    return ((n1 + n2) / 2.0);
}
```

- Υπολογίζει το μέσο όρο 3 αριθμών:

```
double average(double n1, double n2, double n3)
{
    return ((n1 + n2 + n3) / 3.0);
}
```

- Ίδιο όνομα, δυο συναρτήσεις

Παράδειγμα Υπερφόρτωσης: Μέσος Όρος

- Ποια συνάρτηση καλείται;
- Εξαρτάται από το ίδιο το κάλεσμα:
 - `avg = average(5.2, 6.7);`
 - Καλεί "δύο-παραμέτρων `average()`"
 - `avg = average(6.5, 8.5, 4.2);`
 - Καλεί "τριών-παραμέτρων `average()`"
- Ο μεταφραστής επιλύει τέτοια θέματα βασιζόμενος στην υπογραφή στο κάλεσμα
 - "Ταιριάζει" το κάλεσμα με την αντίστοιχη συν/ση

Επίλυση Υπερφόρτωσης

- 1^{ον}: Απόλυτο ταίριασμα
 - Κοιτάζει για ίδια υπογραφή
 - όπου δεν απαιτείται κάποια μετατροπή τύπου
- 2^{ον}: Συμβατό ταίριασμα
 - Κοιτάζει για "συμβατή" υπογραφή όπου μια αυτόματη μετατροπή τύπου είναι δυνατή:
 - 1^{ος} προώθηση (π.χ., int→double)
 - δεν χάνονται δεδομένα
 - 2^{ος} αποκοπή (π.χ., double→int)
 - πιθανή απώλεια δεδομένων
- Ομοιότητες στα ίδια επίπεδα ⇒ σφάλμα στο μεταφραστή

```
void f(int n, double m);  
void f(double n, int m);
```

```
f(33, 44);
```

Αυτόματη μετατροπή τύπου και Υπερφόρτωση

- Οι τυπικοί αριθμητικοί τύποι φτιάχνουν συνήθως τύπους "double"
- Επιτρέπει "οποιοδήποτε" αριθμητικό τύπο
 - `int` → `double`
 - `float` → `double`
 - `char` → `double` *αργότερα θα το δούμε!
- Αποφεύγει υπερφόρτωση για διαφορετικούς αριθμητικούς τύπους

Αυτόματη μετατροπή τύπου και Υπερφόρτωση

- `double kmlgas(double klm, double liters)`
`{`
 `return (klm/liters);`
`}`
- Παραδείγματα :
 - `a = kmlgas(5, 20);`
 - Μετατρέπει 5 & 20 σε doubles, και μετά περνάει τιμές
 - `a = kmlgas (5.8, 20.2);`
 - Δεν χρειάζεται μετατροπή
 - `a = kmlgas (5, 2.4);`
 - Μετατρέπει το 5 σε 5.0, και μετά περνάει τιμές στη συν/ση

Προεπιλεγμένα Ορίσματα

- Κατά την κλήση επιτρέπει την αποφυγή μερικών ορισμάτων
- Στη δήλωση/ορισμό συνάρτησης:

```
void showVolume( int length,  
                int width = 1,  
                int height = 1 );
```

- Τα τελευταία 2 ορίσματα είναι προεπιλεγμένα (defaulted)

– Πιθανά καλέσματα:

- `showVolume(2, 4, 6);` //τιμές σε όλα τα ορίσματα
- `showVolume(3, 5);` //height προεπιλεγμένο σε 1
- `showVolume(7);` //width & height προεπιλεγμένα σε 1

```
#include <iostream>
using namespace std;
```

```
void showVolume(int length, int width = 1, int height = 1);
//Returns the volume of a box.
```

```
int main( )
{
    showVolume(4, 6, 2);
    showVolume(4, 6);
    showVolume(4);

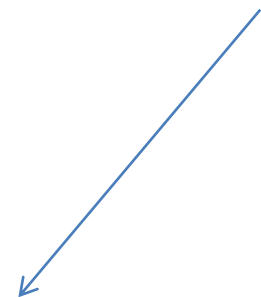
    return 0;
}
```

```
void showVolume(int length, int width, int height)
{
    cout << "Volume of a box with \n"
         << "Length = " << length << ", Width = " << width << endl
         << "and Height = " << height
         << " is " << length*width*height << endl;
}
```

Προκαθορισμένα ορίσματα



Ένα προκαθορισμένο όρισμα δεν πρέπει να δίνεται στο κυρίως σώμα της συν/σης



Συναρτήσεις (σύνοψη)

- Όταν πρόκειται να γράψουμε μια συν/ση:
 1. Καθορίζουμε τον επιστρεφόμενο τύπο
 - Πχ. `int`
 2. Δίνουμε όνομα στην συν/ση
 - Πχ. `square`
 3. Δηλώνουμε τα ορίσματα (τι θα πρέπει να δέχεται η συν/ση ως είσοδο)
 - Πχ. `(int y)`
 4. Γράφουμε την δήλωση της συν/σης
 - Πχ. `int square(int y);`
 5. Υλοποιούμε την μέθοδο έχοντας υπόψιν την εντολή `return`
 - Πχ.

```
int square(int y)
{
    int result;
    result = y*y;
    return result;
}
```

Καλή Μελέτη

- **Βιβλιογραφία**

[1] W. Savitch, Πλήρης C++, Εκδόσεις Τζιόλα, 2011

[2] H. Deitel and P. Deitel, C++ Προγραμματισμός 6η Εκδοση, Εκδόσεις Μ. Γκιούρδας, 2013

Ύλη βιβλιογραφίας

[1]: 3.1, 3.2, 3.3, 4.1, 4.2, 13.1, 13.2

[2]: Κεφ. 6