ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Δρ. Χρήστος Α. Φείδας, Επίκουρος Καθηγητής

Ακαδημαϊκό Έτος 2021-2022

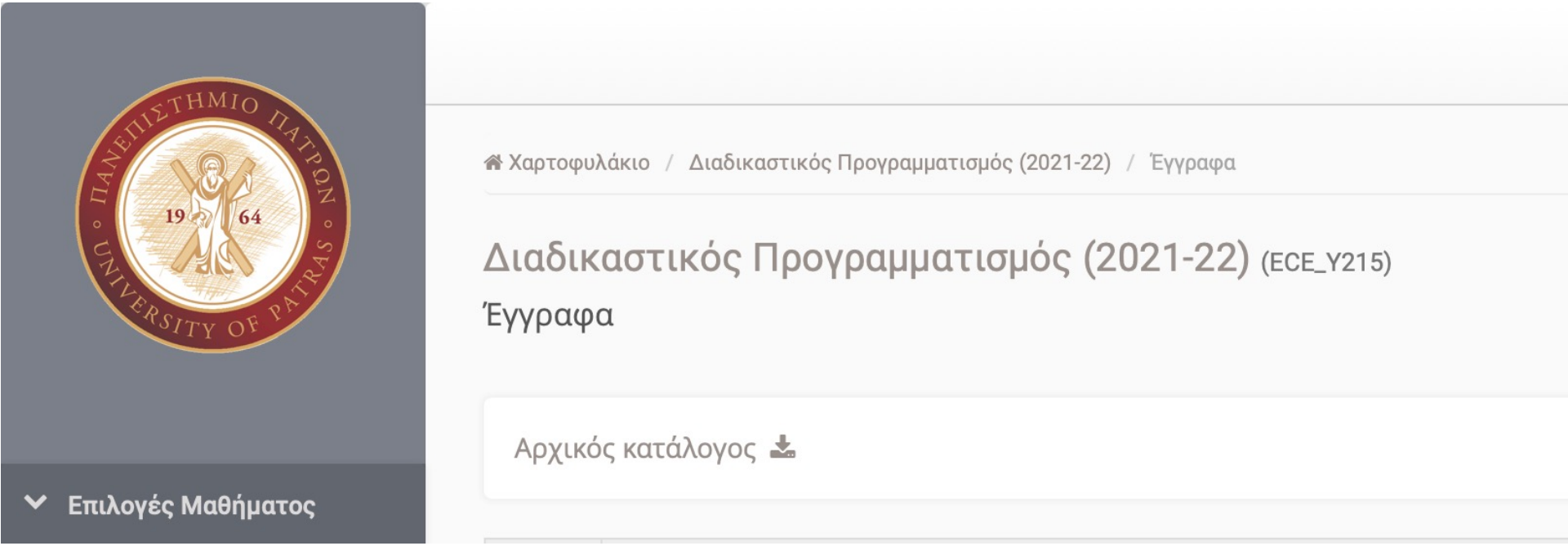
Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών
Πανεπιστήμιο Πατρών

1η Εβδομάδα: Εισαγωγικές Έννοιες

Γενικές Πληροφορίες

Διδάσκων Καθηγητής:	**Χρήστος Φείδας** (skypeID: **fidas.christos**)
Γραφείο:	Εργαστήριο Συστημάτων Υπολογιστών
Τηλέφωνο:	2610996491
E-mail:	**fidas@upatras.gr**
Ιστοσελίδα Μαθήματος:	eclass.upatras.gr → Διαδικαστικός Προγραμματισμός (2021-22)
Διαλέξεις:	**Τετάρτη** 9:00 -11:00 & **Παρασκευή**, 09:00 -10:00
Αίθουσα:	ΗΛ6
Φροντιστήρια:	-
Ώρες γραφείου:	**Δευτέρα 17:00 – 19:00** ή μετά από συνεννόηση
Εργαστήρια:	**Κάθε 2 εβδομάδες** (συνήθως 5 με 6 εργαστηριακές ασκήσεις) **Πρώτο εργαστήριο 9 και 10 Μαρτίου 2022**

Σελίδα Μαθήματος



- **Πληροφορίες**
- **Διαφάνειες**
- **Παράδοση εργασιών**
- **Βαθμολογία**
- **Ανακοινώσεις**
- **Επικοινωνία**

Τι θα μάθουμε;

- Κατανόηση της έννοιας του διαδικαστικού προγραμματισμού
- Θεμελίωση της αλγοριθμικής σκέψης για την επίλυση προβλημάτων με τον προγραμματισμό
- Ικανότητα σχεδιασμού, υλοποίησης, δοκιμής, αποσφαλμάτωσης και τεκμηρίωσης προγραμμάτων στη γλώσσας προγραμματισμού C
- Αξιοποίηση του υλικού ενός υπολογιστικού συστήματος για την δημιουργία προγραμμάτων υψηλής **απόδοσης**
- Την απόκτηση εμπειρικής γνώσης μέσα από τα εργαστήρια

Πως θα το μάθουμε;

- **Διαλέξεις**
  - Συνίσταται παρακολούθηση
  - Παρουσιάζονται οι διάφορες έννοιες
  - Επιδιώκεται ο διάλογος και η ανταλλαγή επιχειρημάτων
- **Εργαστήρια**
  - Υποχρεωτική παρακολούθηση
  - Παρουσίαση εργαλείων
  - Εμπέδωση εννοιών σε καθαρά πρακτικό επίπεδο
- **Ασκήσεις για την καλύτερη προετοιμασία των εργαστηρίων**
  - Συνίσταται ενεργός συμμετοχή
  - Εφαρμογή εννοιών προγραμματισμού για την επίλυση προβλημάτων σε γλώσσα προγραμματισμού C

Αξιολόγηση και Βαθμολόγηση

Εργαστήριο	50%
Τελική Γραπτή Εξέταση	50%

Η επιτυχή ολοκλήρωση του εργαστηρίου είναι απαίτηση για τη συμμετοχή σας στην τελική γραπτή εξέταση του μαθήματος

Απαραίτητες προϋποθέσεις επιτυχίας στο εργαστήριο είναι:

- **Επαρκής παρουσίες του φοιτητή στα εργαστήρια**
- **Τελικός βαθμός από εργαστηριακές ασκήσεις >=5**

Τεκμηρίωση Πιστωτικών Μονάδων

Ώρες εργασίας του μέσου φοιτητή/φοιτήτριας για την επίτευξη μαθησιακών αποτελεσμάτων

5 ΠΜ = 150-180 ώρες εργασίας

Παρακολούθηση διαλέξεων : (3 ώρες x 13 εβδομάδες)	39 ώρες
Παρακολούθηση εργαστηρίων: (2 ώρες X 6 εβδομάδες)	12 ώρες
Προετοιμασία για τα εργαστήρια: (4 ώρες X 6 εβδομάδες)	24 ώρες
Μελέτη για τις διαλέξεις: (4 ώρες x 13 εβδομάδες)	52 ώρες
Σύνολο ωρών:	**127 ώρες**

Αναφορές σχετικά με το Υλικό Παραδόσεων

Οι διαφάνειες εν μέρει στηρίζονται σε υλικό παραδόσεων παλαιότερων ετών του **Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογία Υπολογιστών του Πανεπιστημίου Πατρών** καθώς και του **Τμήματος Πληροφορικής του Πανεπιστήμιο Κύπρου**

Υπολογιστές και Προγραμματισμός

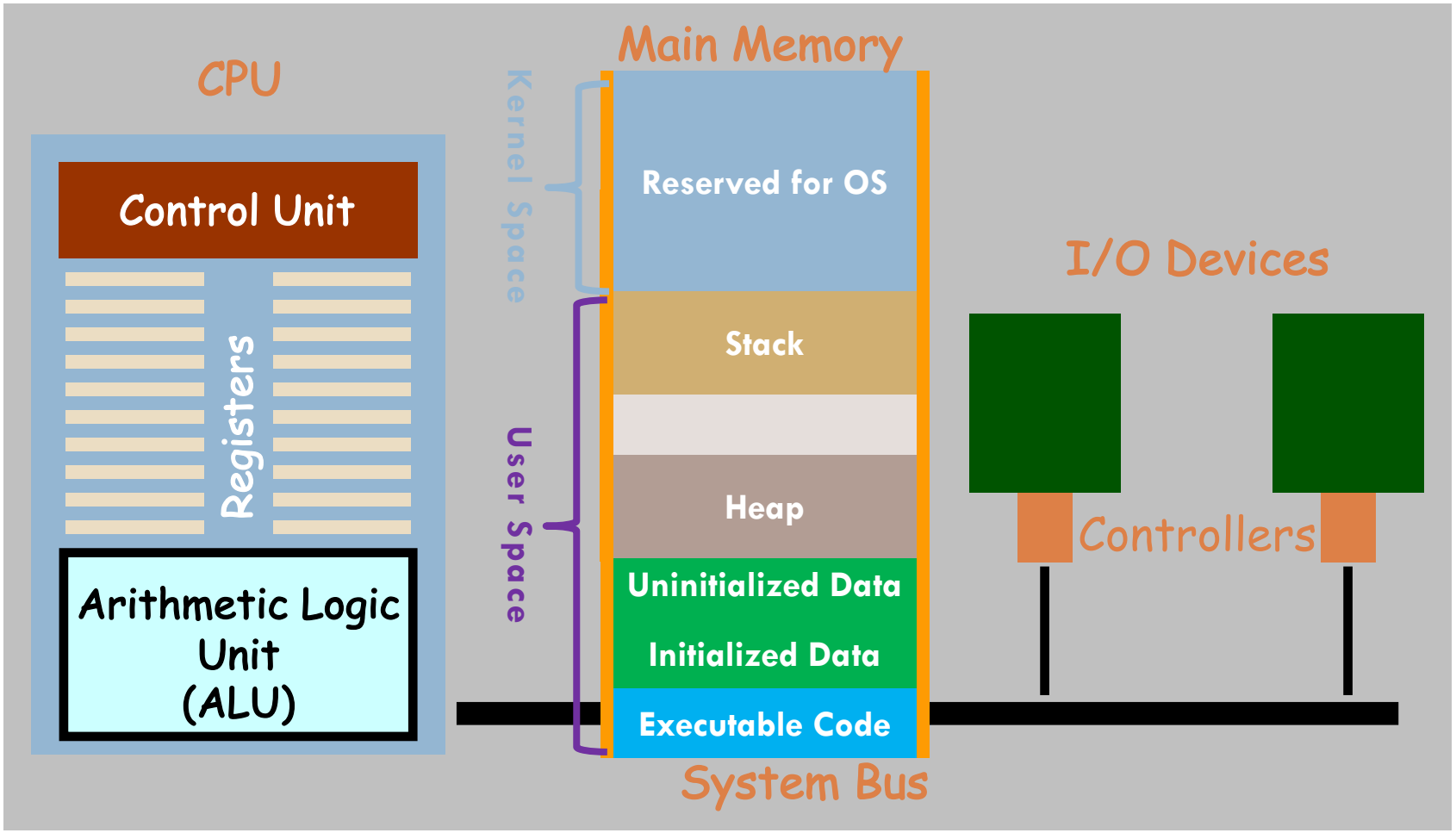
Βασικοί - Ορισμοί

- Αλγόριθμος
  - Μια ταξινομημένη ακολουθία από μη διφορούμενα **βήματα** που οδηγούν στη **λύση** ενός προβλήματος.
- Πρόγραμμα
  - **Αναπαράσταση** ενός ή πολλών αλγορίθμων σε μορφή **κατανοητή από τον υπολογιστή.**
- Προγραμματισμός
  - Ο **σχεδιασμός αλγορίθμων** και η **υλοποίησή τους σε γλώσσες προγραμματισμού** για την επίλυση προβλημάτων.

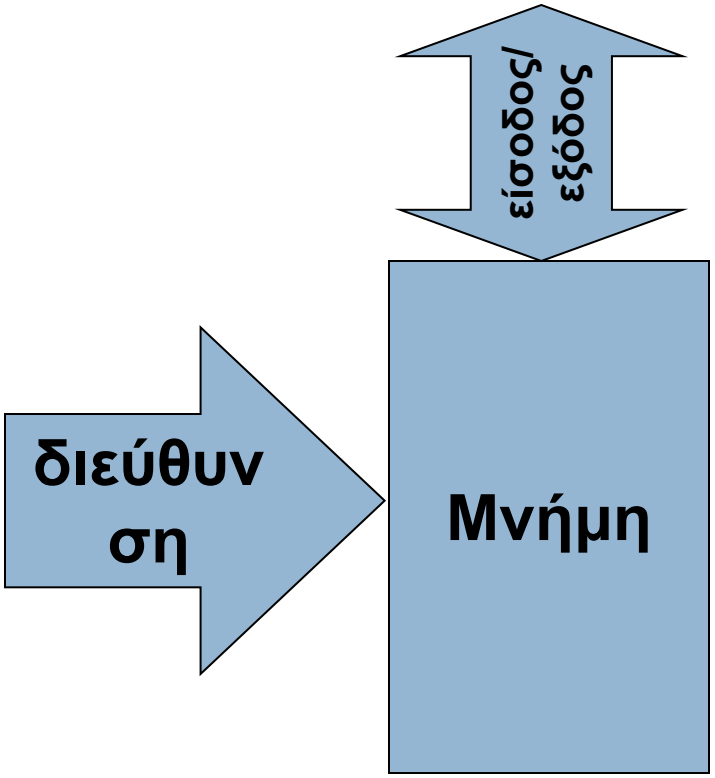
Βασικοί - Ορισμοί

- Γλώσσα Προγραμματισμού
  - Το σύνολο των γραμματικών και συντακτικών κανόνων που μας επιτρέπει να δίνουμε εντολές στον Η/Υ μέσω ενός προγράμματος.
- Κύκλος Ανάπτυξης Προγράμματος
  - Η διαδικασία που ακολουθούμε για την ανάπτυξη ενός προγράμματος.

Ο Υπολογιστής



Η Μνήμη



Διεύθυνση (address)	Περιεχόμενα
0x0000	00111111
0x0001	10110101
0x0002	11011110
0x0003	10111100
…	
0x00FF	00111101
0x0100	10101110
…	

Περιεχόμενα:
Δεδομένα και εντολές

Πόσα bytes διαθέτονται για κάθε μεταβλητή;

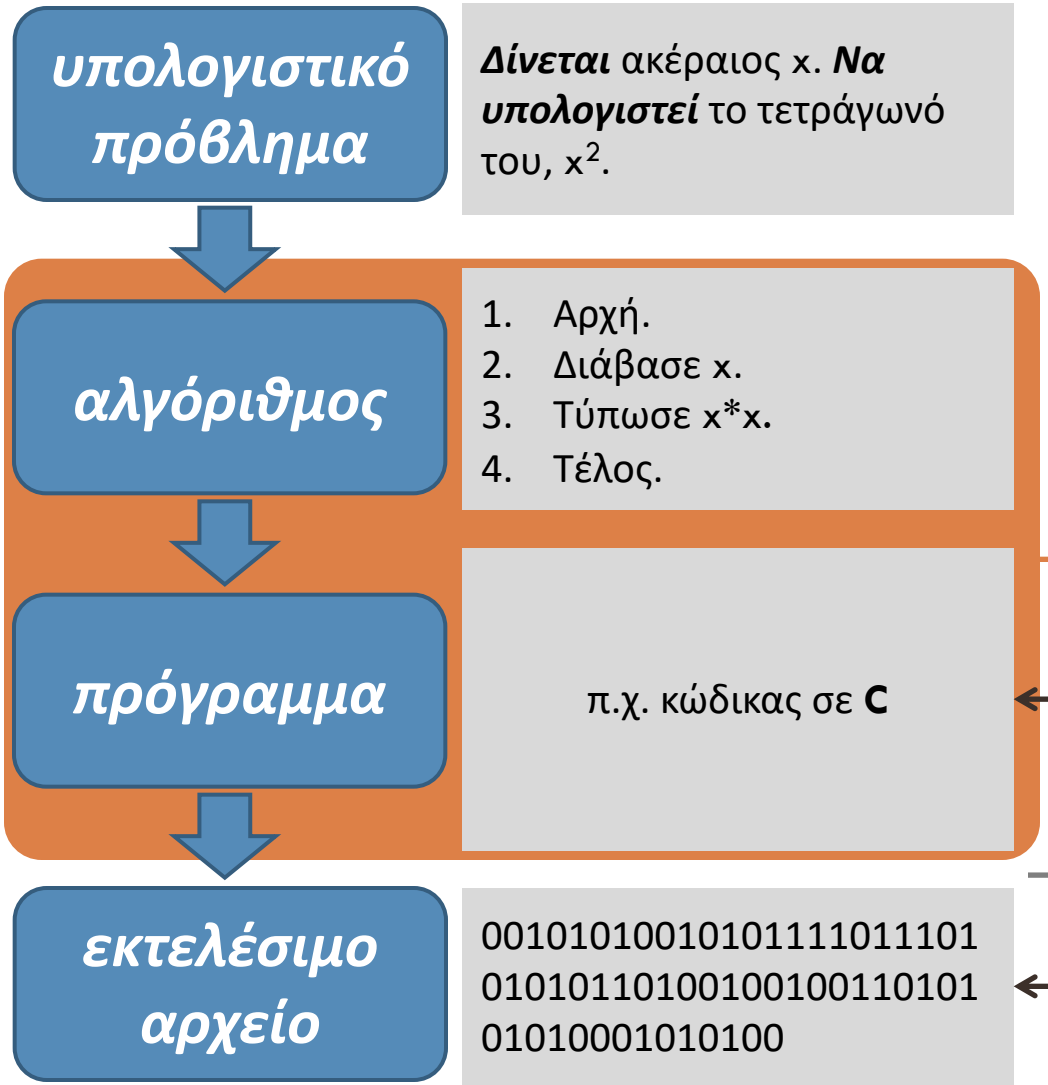
Μεταφραστές (Compilers)

Μεταφραστές - Ορισμός: Μεταφράζουν τις γλώσσες ψηλού επιπέδου σε κώδικα μηχανής

- Πηγαίο Πρόγραμμα (Source code)
  - το αρχικό πρόγραμμα σε **γλώσσα ψηλού επιπέδου**
- Εκτελέσιμο Αρχείο (Executable)
  - το μεταφρασμένο πρόγραμμα σε **γλώσσα μηχανής**
- Στο εργαστήριο θα το κάνουμε με τον πιο κάτω τρόπο:

gcc myprogram.c –o myprogram.exe

Προγραμματισμός



Προγραμματισμός: η διαδικασία του να επινοήσουμε αλγόριθμο και να συντάξουμε πρόγραμμα.

Πρόγραμμα: αλγόριθμος σε *τυπική* μορφή

Μεταγλώττιση (*compilation*)

Εκτελέσιμο αρχείο: αλγόριθμος σε μορφή κατανοητή από τον Η/Υ.

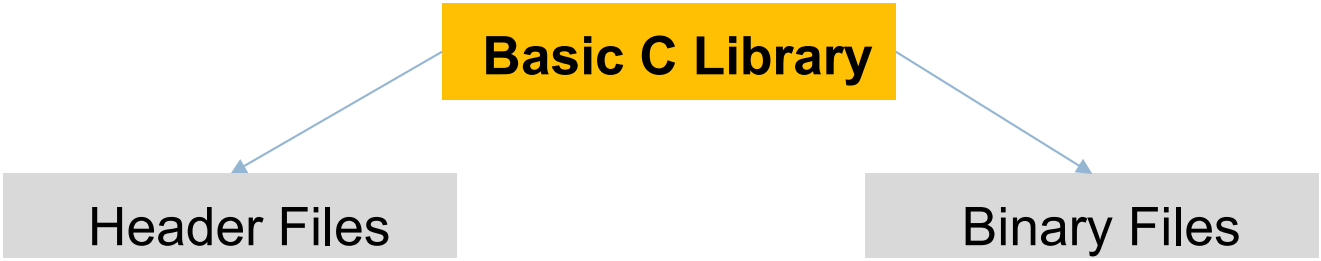
Η γλώσσα C ως αφαίρεση

- Οι γλώσσες υψηλού επιπέδου είναι μια μορφή αφαίρεσης.
  - Η πολυπλοκότητα του **κώδικα μηχανής** αποκρύπτεται από τον προγραμματιστή.
- Ο μεταγλωττιστής (compiler) αντιστοιχίζει στον κώδικα C σύνολα εντολών κώδικα μηχανής

Βιβλιοθήκες στη C

Βιβλιοθήκη είναι μια συλλογή μεταγλωττισμένων μονάδων που μπορούν να συνδεθούν (linked) στα προγράμματά μας μέσω των διεπαφών (header files - interfaces) που παρέχουν.

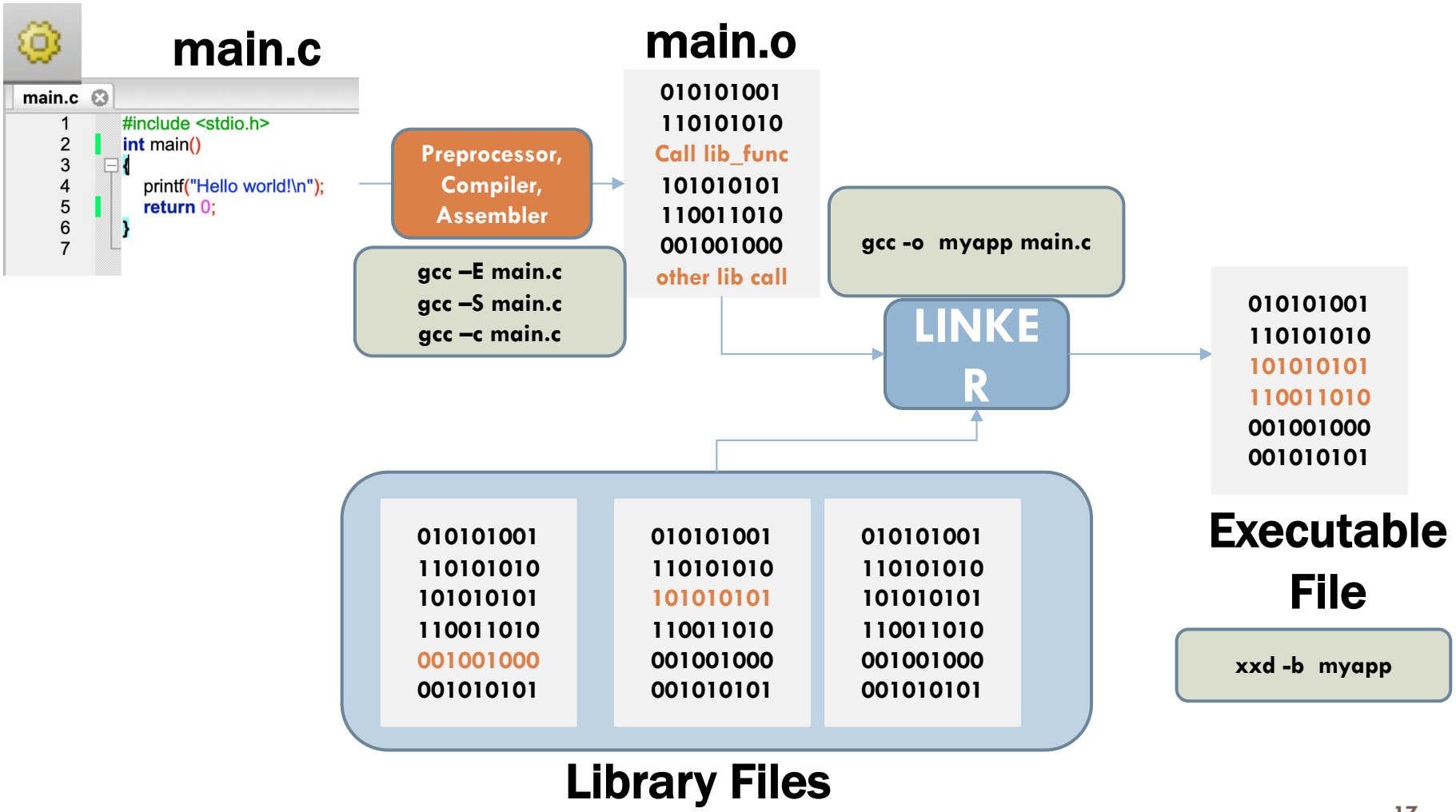
Με άλλα λόγια κάθε βιβλιοθήκη **αποτελείτε από** δυαδικά αρχεία σε object code (*.o) που περιέχουν υλοποιήσεις όλων των συναρτήσεων που έχουν δηλωθεί στα αρχεία επικεφαλίδων .h



- `<stdio.h>` **είναι η διεπαφή (interface) που περιέχει συναρτήσεις I/O.**

Πηγαίος κώδικας της βασικής βιβλιοθήκης: http://ftp.gnu.org/gnu/glibc/

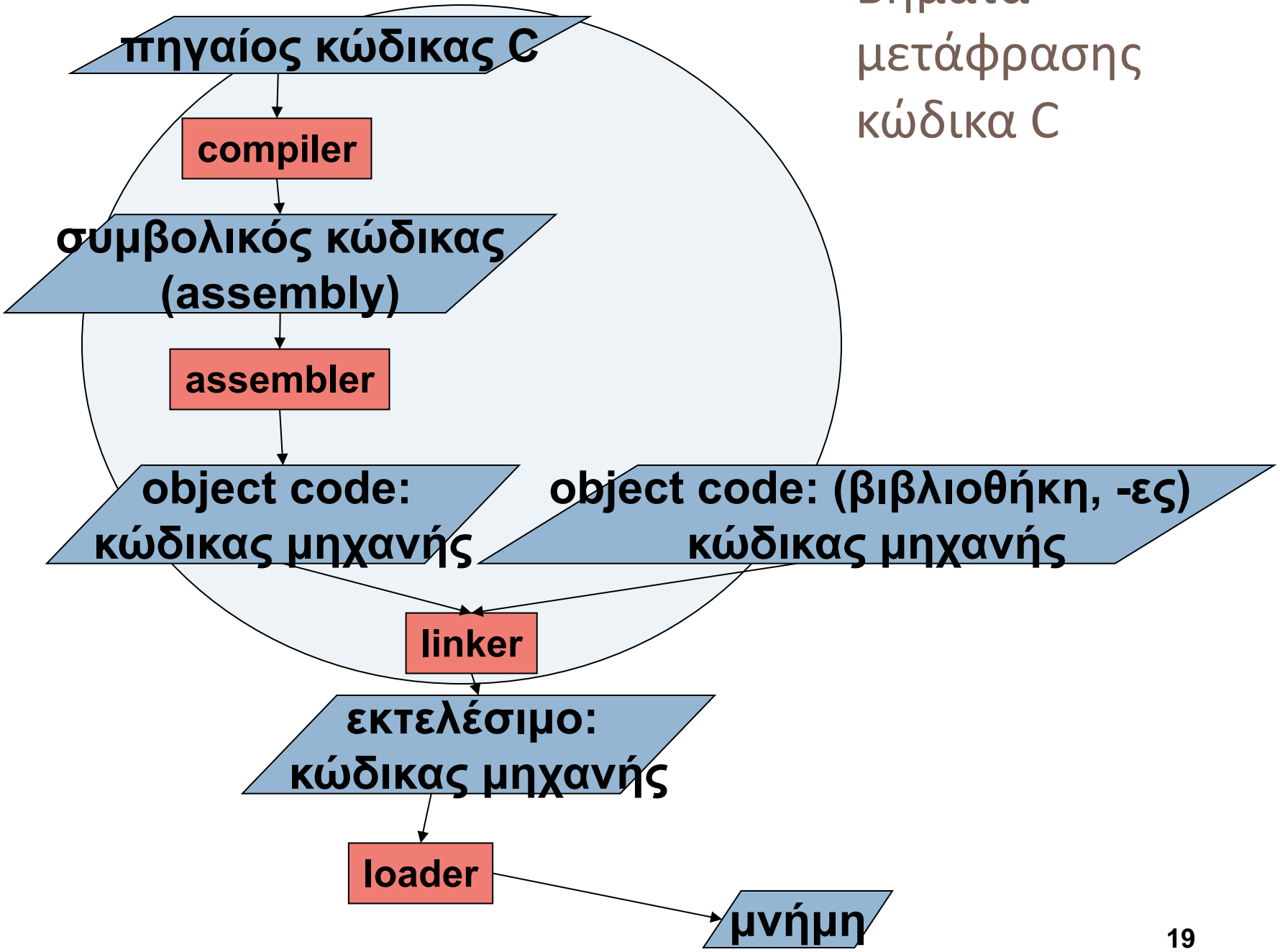
Βιβλιοθήκες στη C



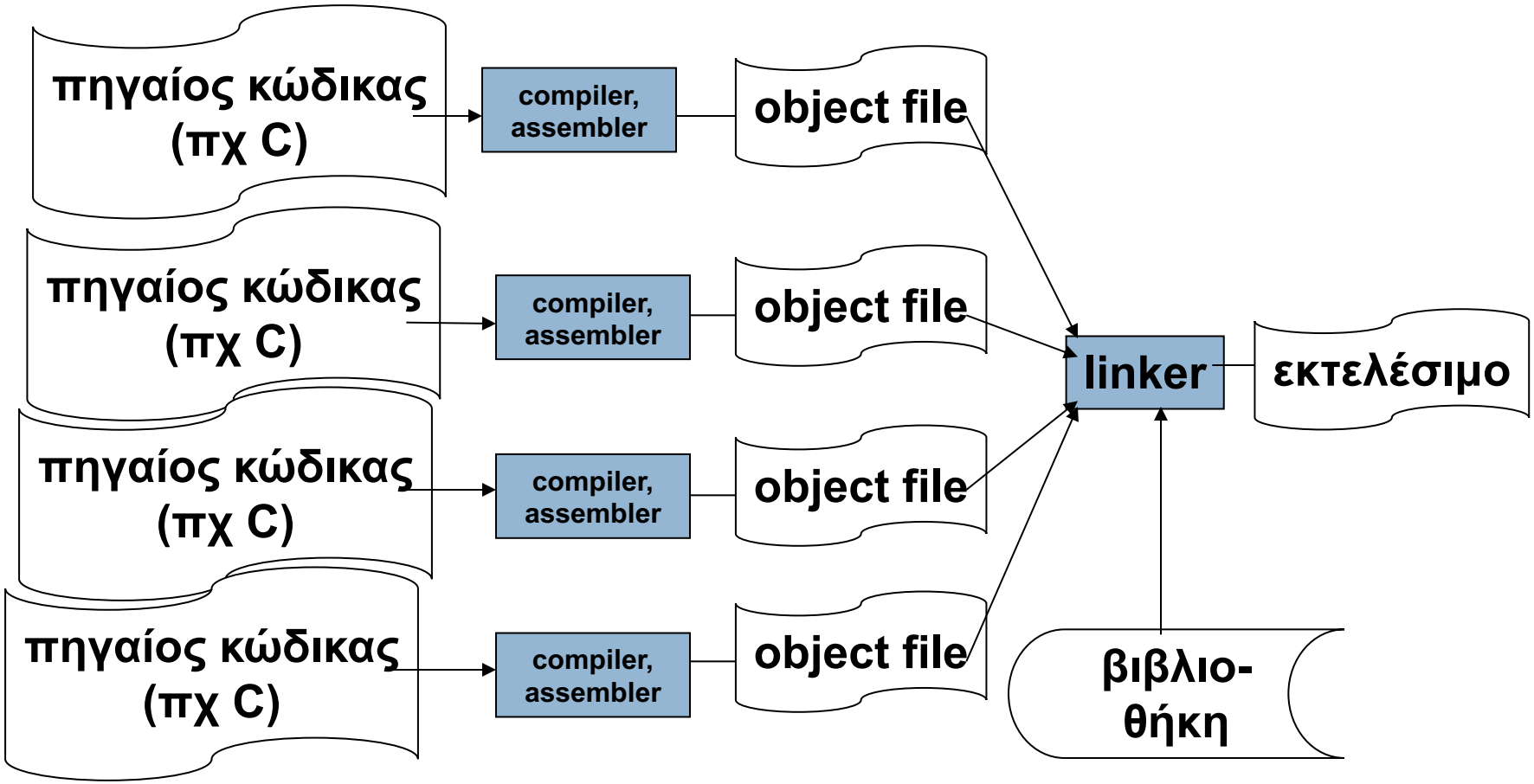
Ενσωμάτωση βιβλιοθηκών (συν.)

- Ενσωμάτωση
  - επιτρέπει την χρήση συναρτήσεων και σταθερών μιας βιβλιοθήκης
  - π.χ. η βιβλιοθήκη stdio περιέχει, ανάμεσα σε άλλα, τις συναρτήσεις:
    - printf, για εκτύπωση πληροφοριών, και
    - scanf, για εισδοχή πληροφοριών
- Άλλες διαταγές στον προεξεργαστή:
  - #define, #ifdef, #if, #else, #endif, #undef

Βήματα μετάφρασης κώδικα C



Από γλώσσα υψηλού επιπέδου σε εκτελέσιμο



Αναγνωριστές στην C

- Κανόνες ονοματολογίας (σταθερές, μεταβλητές, συναρτήσεις):
  - Τα ονόματα μεταβλητών μπορούν να περιέχουν *γράμματα*, *ψηφία*, και _
  - Τα ονόματα μεταβλητών **πρέπει** να ξεκινάνε με γράμματα
  - Η C είναι **CASE SENSITIVE**
    - *κεφαλαία* και *μικρά* γράμματα είναι διαφορετικά!
    - παράδειγμα: int foo; και int FOO; είναι δυο διαφορετικά ονόματα
  - **Απαγορεύεται** να χρησιμοποιούμε σαν ονόματα τις δεσμευμένες λέξεις (βλέπε επόμενη διαφάνεια)

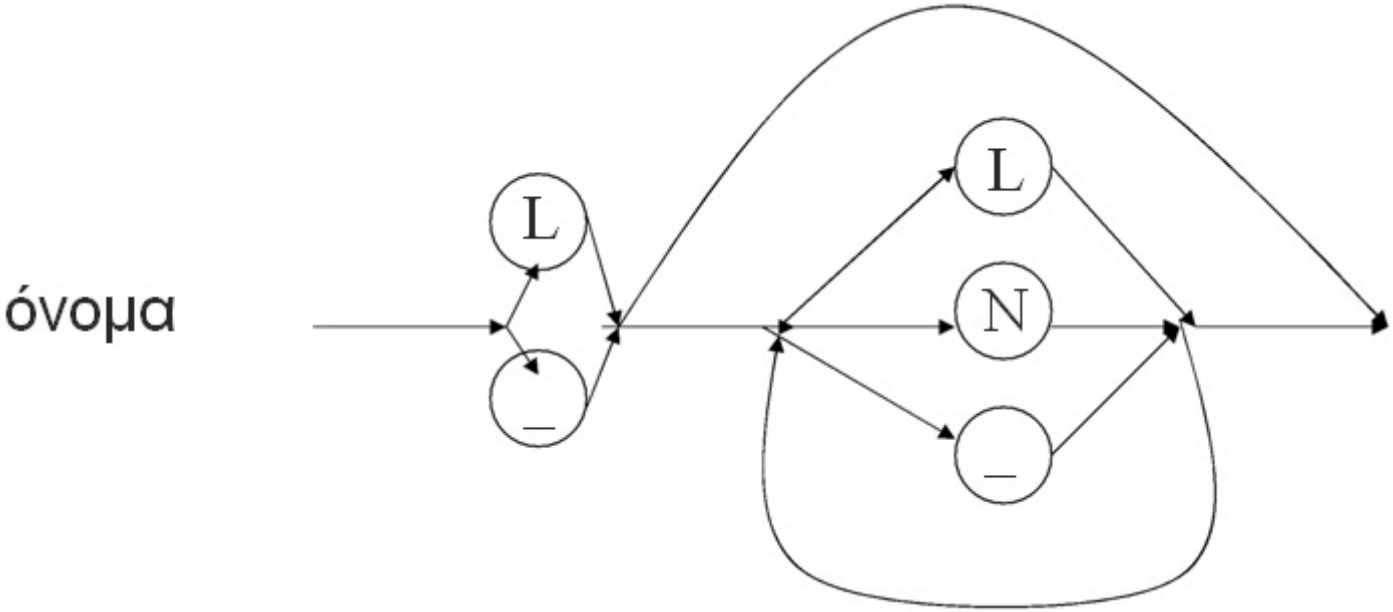
- Καλή Πρακτική: Να χρησιμοποιείτε αυτοεπεξηγηματικά ονόματα

Δεσμευμένες Λέξεις (Reserved Words)

- **Λέξεις με ειδική σημασία για την C**
  - Δεν πρέπει να ορίζονται ξανά σαν ονόματα

auto	**double**	**int**	**struct**
break	**else**	**long**	**switch**
case	**enum**	**register**	**typedef**
char	**extern**	**return**	**union**
const	**float**	**short**	**unsigned**
continue	**for**	**signed**	**void**
default	**goto**	**sizeof**	**volatile**
do	**if**	**static**	**while**

Σύνταξη Αναγνωριστών



L: letters a-z A-Z
N: numbers 0-9
_ : underscore

Παραδείγματα (σωστό η λάθος;):

- **int metavliti;** (σωστό)
- **int rectangle_area;** (σωστό)
- **int _index_123;** (σωστό)
- **int 1variable;** (λάθος)
- **int %super^;** (λάθος)
- **int se tria meroi;** (λάθος)

ΜΕΤΑΒΛΗΤΕΣ

Σημασία Μεταβλητής (variable)

- Αποθήκευση και ανάγνωση τιμών
  - Αριθμών, χαρακτήρων, κ.ά
- Κάθε μεταβλητή χαρακτηρίζεται από:
  - **τύπο**
    - βασικοί: int, char, float, double
    - μέγεθος: 4B, 1B, 4B, 8B (κάθε κυψελίδα 1 Byte)
  - **(συμβολικό) όνομα**
    - το όνομα αντιστοιχεί σε συγκεκριμένη διεύθυνση στην μνήμη
  - **τιμή**
- Μια μεταβλητή πρέπει να **δηλωθεί** πριν χρησιμοποιηθεί

Εξαρτάται από το λειτουργικό σύστημα

Σημασία Μεταβλητής (συν.)

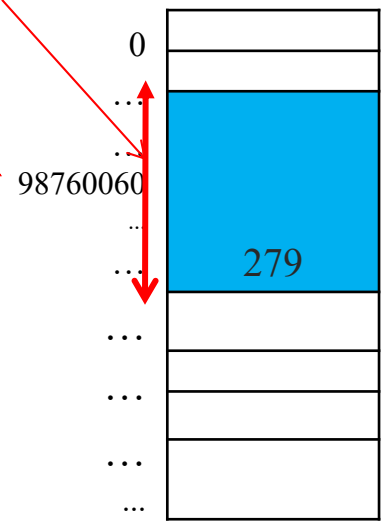
τύπος

Όνομα (διεύθυνση)

τιμή

```
int count;

count = 137;

count = count + 142;

foo(count)
foo(&count);
```

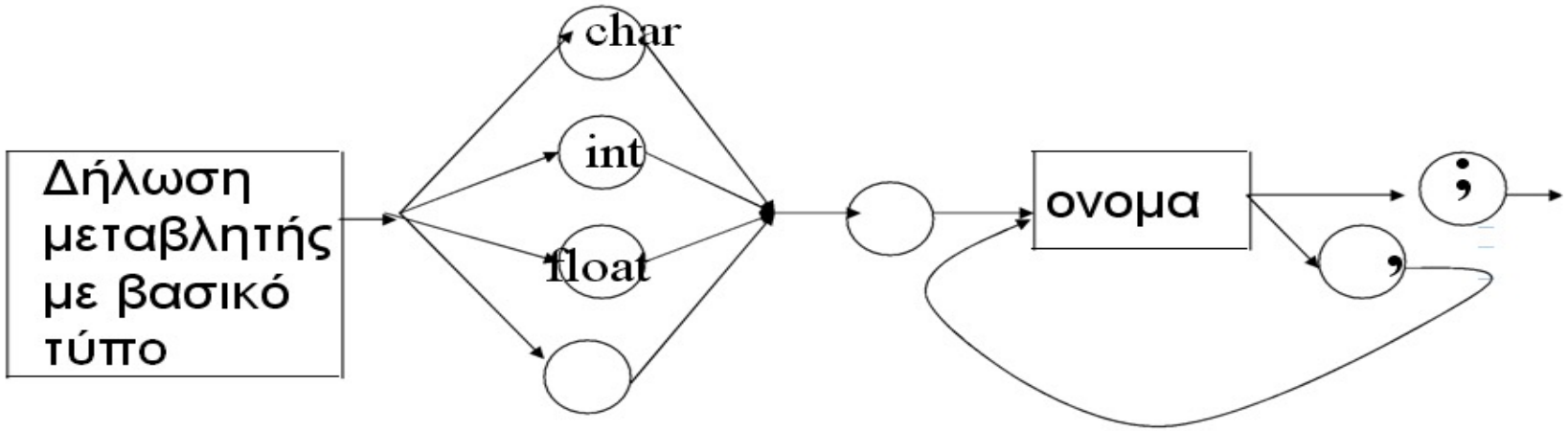
Τιμή του count



Διεύθυνση του count

Τύπος διεύθυνσης ονομάζεται δείκτης (pointer) π.χ. int *deikths;

Δήλωση Μεταβλητής



Παραδείγματα (σωστό η λάθος?):

- **char $foo;** (σωστό)
- **char ____f;** (σωστό)
- **int count, sum;** (σωστό)
- **float area, double volume;** (λάθος)
- **int index; cost;** (λάθος)
- **double charge; float angle, income;** (σωστό)

Σημασία Σταθεράς (constant)

- Σταθερές έχουν
  - τιμή
  - όνομα όχι όμως διεύθυνση
  - προ-επεξεργαστής αντικαθιστά το όνομα με την τιμή
- Διευκολύνει
  - τροποποιήσεις
  - διάβασμα ενός προγράμματος (PI ή 3.1453)
- Μια σταθερά πρέπει να δηλωθεί πριν χρησιμοποιηθεί
- **Δεν μπορεί να αλλάξει τιμή κατά την εκτέλεση του προγράμματος**

Δήλωση Σταθεράς

- Δήλωση Σταθεράς: #define → [όνομα] → [τιμή]

- #define PI 3.1453
- #define YES 1
- #define NO 1
- #define TRUE 1
- #define PISQUARE PI*PI
- #define ERROR_SPACE_MESSAGE “error:run out of space”

Τύποι δεδομένων

Ένας τύπος δεδομένων είναι ένα σύνολο τιμών και ένα σύνολο λειτουργιών (πράξεων) που μπορούν να εφαρμοστούν σε αυτές τις τιμές

- Βασικοί τύποι δεδομένων
  - char, int, float, double
- Σύνθετοι τύποι δεδομένων: (πίνακες, δομές)

int (ακέραιος)

- Αναπαριστά ακεραίους αριθμούς
- Μέγεθος 4 bytes,πεδίο τιμών -2^{31}.. $+2^{31}-1$
  - (≈ -2 δις – 2 δις)
- Παραδείγματα κυριολεκτικών τιμών:
  - -2189456 0 50 +24562 –3245 13576313
- Πράξεις: πρόσθεση (+), αφαίρεση (-), πολλαπλασιασμός (*), διαίρεση (/), υπόλοιπο (%) ….

float (κινητής υποδιαστολής μονής ακριβείας)

- Αναπαριστά τους πραγματικούς αριθμούς
- Μέγεθος: 4 bytes, $1x10^{38}$,$1x10^{-37}$
- Πράξεις: πρόσθεση (+), αφαίρεση (-), πολλαπλασιασμός (*), διαίρεση (/)
- Σημειογραφία για κυριολεκτικές τιμές:
  - 1.258×10^{6} = **1258000.0** = **1.258e6** = **1.258E6**
  - 8.9×10^{-4} = **0.00089** = **8.9e-4** = **8.9E-4**

double (κινητής υποδιαστολής διπλής ακριβείας)

- Ίδιος τύπος με float αλλά με μεγαλύτερη ακρίβεια
- Μέγεθος 8 bytes

Valid double Constants	Invalid double Constants
3.14159	150 (no decimal point)
0.005	.12345e (missing exponent)
12345.0	15e-0.3 (0.3 is invalid exponent)
15.0e-04 (0.0015)	
2.345e2 (234.5)	12.5e.3 (.3 is invalid exponent)
1.15e-3 (0.00115)	34,500.99 (comma is not allowed)
12e+5 (1200000.0)	

Χαρακτήρες (char)

- Χειρισμός χαρακτήρων (ατομικών)
  - A-Z, a-z, 0-9, !@$%&#, ειδικά σύμβολα \n
  - Μέγεθος 1 byte
  - Εσωκλείονται σε απλές (μονές) αποστρόφους
    - ′A′, ′a′, ′9′, ′″′, ′ ′, ′*′, ′\n′, ′\′′, κτλ
    - Κατά την είσοδο από το πληκτρολόγιο δε χρειάζονται απόστροφοι
- Παραδείγματα
  - ´a´ είναι ο χαρακτήρας a
  - ´b´ είναι ο χαρακτήρας b
  - ´9´ είναι ο χαρακτήρας 9
  - ´*´ είναι ο χαρακτήρας *

Χαρακτήρες

- Αντιστοιχούν σε ένα μοναδικό κωδικό
- Στη C ο κωδικός αναφέρεται στον πίνακα ASCII
  - Αλφαβητικοί, ψηφιακοί, ειδικοί (\n, \t)
  - ‘0’ ascii:48, ‘1’ ascii:49,.., ‘9’ ascii:57
  - ‘A’ ascii:65, …, ‘Z’ ascii:90
  - ‘a’ ascii:97, …, ‘z’ ascii:122

Πίνακας ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Spa	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DE

Οι τύποι δεδομένων περιληπτικά

Τύπος	Μέγεθος	Πεδίο Τιμών
char,	1byte	‘α’..‘z’ ‘A’..‘Z’‘0’..‘9’
int,	4 bytes	$-2^{31}..2^{31}$
float,	4 bytes	$10^{-37}..10^{38}$
double	8 bytes	$10^{-307}..10^{308}$

ΕΙΣΟΔΟΣ / ΕΞΟΔΟΣ

- **printf**
- **scanf**

Εντολές printf και scanf

- Παράδειγμα

```
printf(“That equals %f kilometers and %e meters.\n”, kms, mtrs);
```

- Παράδειγμα scanf

```
scanf(“%d%f”,&arithmos1,&arithmos2);
```

Τύπος δεδομένων	printf	scanf
float	%f	%f
double	%f ή %e	%lf
int	%d	%d
char	%c	%c

Είσοδος / Έξοδος

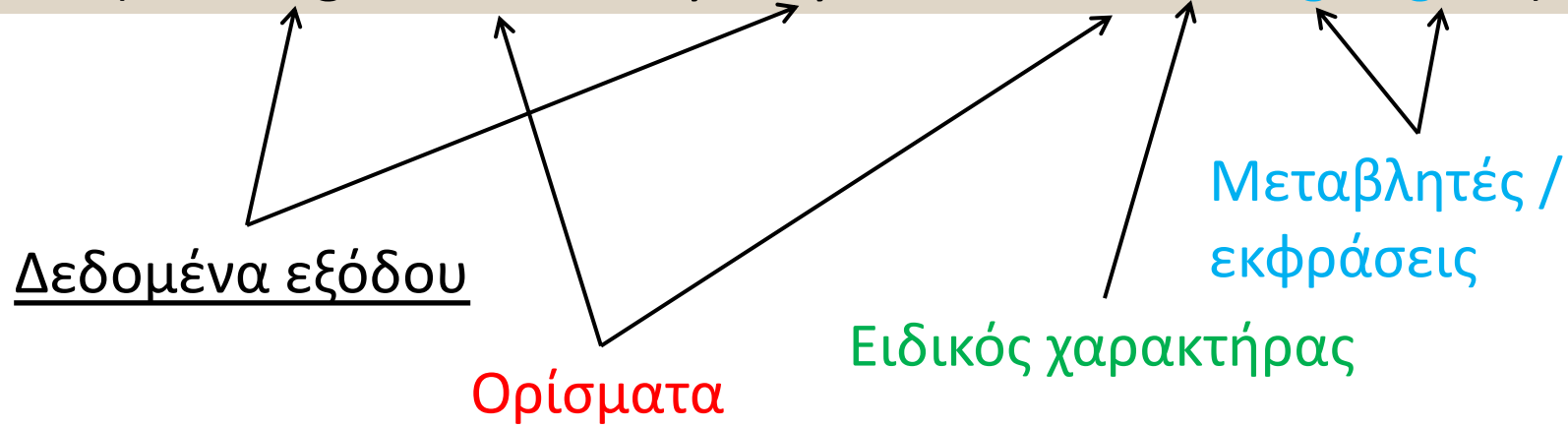
Σχεδόν όλα τα προγράμματα περιλαμβάνουν ανταλλαγή δεδομένων με τον χρήστη. Δηλαδή, είσοδο δεδομένων και έξοδο (εκτύπωση) δεδομένων.

- Συνάρτηση εισόδου – αντιγράφει δεδομένα από μονάδα εισόδου σε χώρο στη μνήμη, (scanf)
- Συνάρτηση εξόδου – εκτυπώνει σε μονάδα εξόδου πληροφορίες αποθηκευμένες στην μνήμη, (printf)
- Η βιβλιοθήκη stdio περιέχει βασικές συναρτήσεις εισόδου/εξόδου

Εντολή printf

- Σύνταξη
  - printf(“μορφή εξόδου”);
  - printf(“μορφή εξόδου”, λίστα εκφράσεων);

```
int age= 29;
printf(“Your age is %d. Next year you will be %d\n”,age,age+1);
```



Δεδομένα εξόδου - Σύνταξη

- Ξεκινούν με “ και τελειώνουν με ”
- Περιλαμβάνουν
  - Αλφαριθμητικά (σειρές από χαρακτήρες)
  - Ορίσματα
  - Ειδικούς χαρακτήρες

Ορίσματα **printf**

- Σύνταξη %d, %c, %f, %e, %s
- Αντικατέστησε το όρισμα με τιμή έκφρασης από τη λίστα εκφράσεων
  - %d ακέραια τιμή (int)
  - %f πραγματικοί (double και float)
  - %e πραγματικοί (επιστημονική γραφή)
  - %c ένας χαρακτήρας (char)
  - %s σειρά από χαρακτήρες (string)

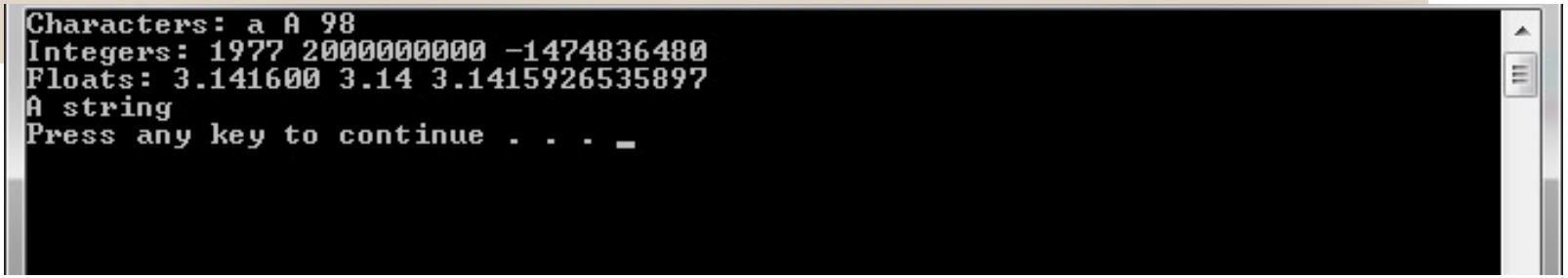
Ο αριθμός των ορισμάτων πρέπει να είναι ίδιος με το μέγεθος της λίστας εκφράσεων και οι τύποι τους να είναι ένας προς ένας αντίστοιχοι

Παραδείγματα


```
printf("To   mathima exei %d foitites\n", 50);

int n_students = 50;
printf("to mathima exei %d foitites\n", n_students);

float average=23.5;
printf("o mesos oros einai %f\n", average);

printf("simeio = (%f, %f, %f)", x, y, z);
```

Παραδείγματα

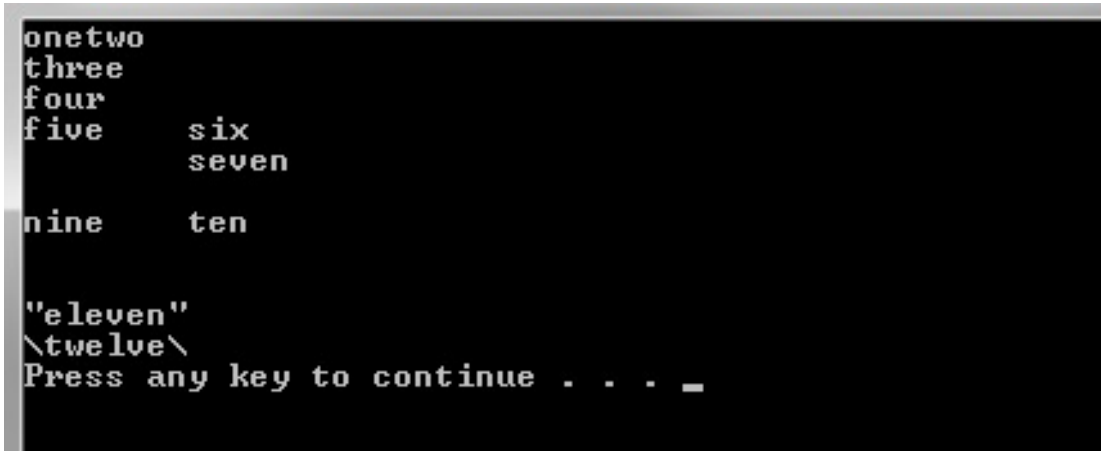

```
#include <stdio.h>
int main()
{
   printf ("Characters: %c %c %d\n", 'a', 65,'b');
   printf ("Integers: %d %d %d\n", 1977, 2000000000,20000000000);
   printf ("Floats: %f %.2f %.13f\n", 3.1416, 3.1416,
   3.1415926535897);
   printf ("%s \n", "A string");
   system("pause");
   return 0;
}
```



Ειδικοί χαρακτήρες

- \n : μετέφερε το δρομέα στην επόμενη γραμμή
- \t : μετέφερε το δρομέα στην επόμενη δεξιά στήλη
- \’’ : εκτύπωσε ’’
- \\ : εκτύπωσε \
- \a ήχησε κουδούνι

```
#include <stdio.h>
int main()
{
   printf ("one");
   printf ("two\n");
   printf ("three\nfour\n");
   printf ("five\tsix\n");
   printf ("\tseven\n\n");
   printf ("nine\tten\n\n\n");
   printf ("\"eleven\" \n");
   printf ("\\twelve\\ \n");
    system("pause");
    return 0;
}
```



Παραδείγματα ορισμάτων

- Πράξεις κατευθείαν στην έκφραση

```
int a = 1, b = 2, c=4;
printf("%d, %d, %d, %d, %d", a, b, a+b, a/b, a*b+c-a);
```

⇒ **Εκτύπωση: 1, 2, 3, 0, 5**

- Αυτόματη μετατροπή (casting)

```
int a = 1, b = 2;
printf(“%d, %d, %f”, a, b, (float)a/b);
```

=> 1, 2, 0.500000

- Ακέραιος ή χαρακτήρας;

```
int a = ‘1’ ;

printf(“%d”, a);
```

=> παίρνει την τιμή που αντιστοιχεί στον χαρακτήρα ’1’ με βάση τον ASCII, δηλαδή 49

=> 49

Μορφοποίηση (formatting) Ακεραίων

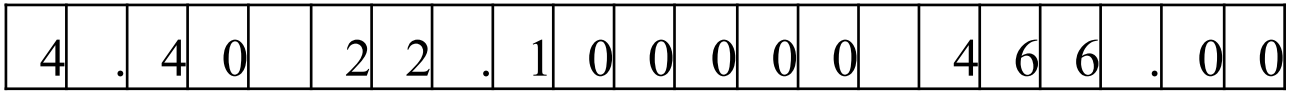
- Σύνταξη: **%nd**
- Σημασία:
  - χρησιμοποίηση **n** θέσεων για την εκτύπωση του ακέραιου αριθμού.
  - Εάν ο αριθμός έχει λιγότερα από **n** ψηφία, τοποθετούνται κενά στα αριστερά (δεξιά ευθυγράμμιση). Εάν ο αριθμός έχει περισσότερα ψηφία, εκτυπώνονται **όλα!**
- printf("\n%3d%3d\n\n%2d%3d\n", 1, 22, 33, -444);

		1		2	2		
3	3	-	4	4	4		

Μορφοποίηση Κινητής Υποδιαστολής

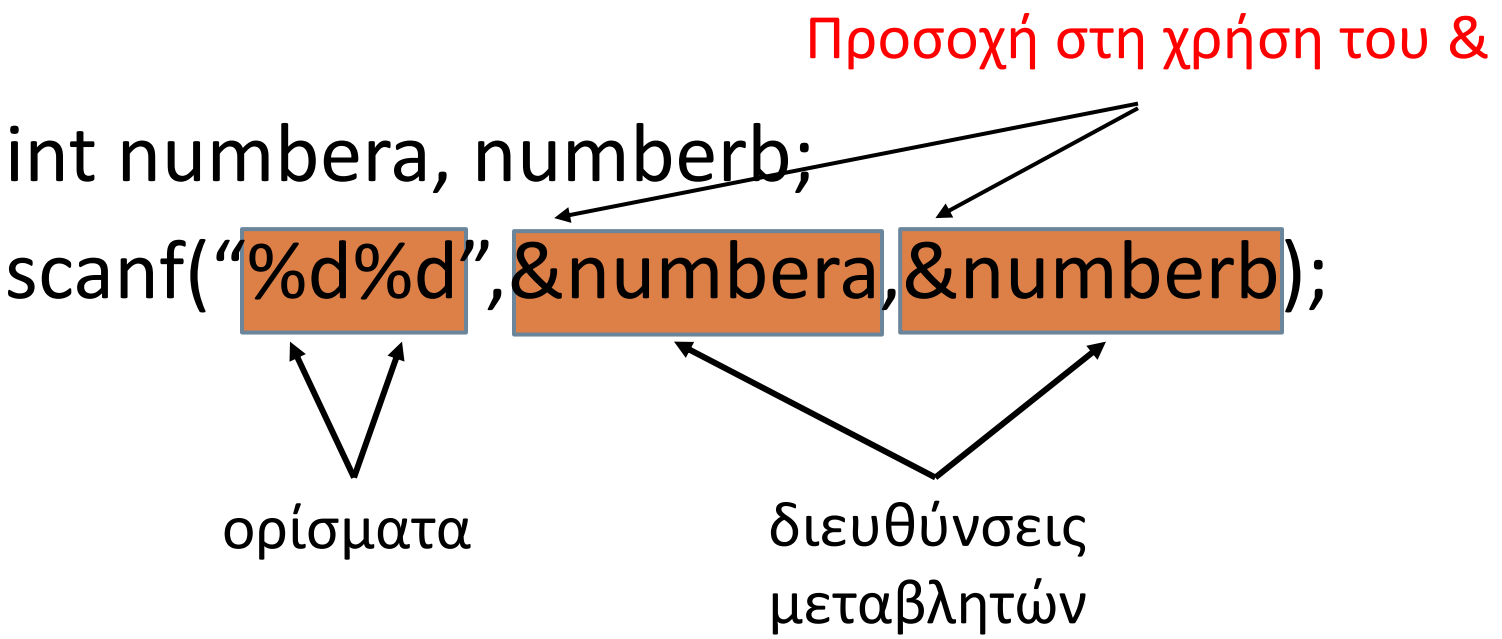
- Σύνταξη: %n.mf
- Σημασία:
  - χρησιμοποίηση *τουλάχιστον* **n** θέσεων για την εκτύπωση του αριθμού (συμπεριλαμβάνει την ΄.΄ και το ΄–΄ ,αν χρειάζεται)
  - Τα m ψηφία να είναι δεκαδικά
  - Εάν ο αριθμός έχει περισσότερα ψηφία, εκτυπώνονται **όλα**
- Σύνταξη: %.mf (m δεκαδικά ψηφία)

printf(”%4.2f %9.6f %3.2f\n”, 4.4, 22.1, 466.00);



Είσοδος – scanf

- Σύνταξη:
  - scanf(μορφή εισόδου, λίστα διευθύνσεων μεταβλητών);



Μορφή Εισόδου

- Σύνταξη:
  - Ξεκινά με “ και τελειώνει με ”
  - Περιλαμβάνει: ορίσματα
- Σημασία:
  - τύπος ορισμάτων και σειρά τιμών που θα εισαχθούν
  - τύποι ανάλογοι με printf (πχ %d %f %e %c κτλ)
  - %d, %f και %e διαπερνούν white space

Τελεστής διεύθυνσης και scanf

- Σύνταξη: &<όνομα μεταβλητής>
- Σημασία: δίνει την διεύθυνση της μεταβλητής

  π.χ. &numbera

- **scanf**: αποθηκεύει τιμές στις μεταβλητές χρησιμοποιώντας τις διευθύνσεις τους

- Όπως κάθε συνάρτηση στην C
  - Θα το δούμε σε λεπτομέρεια πιο μετά (Ενότητα Συναρτήσεων)

Χρήση printf και scanf

```
int numbera, numberb;

printf(“Enter two integer numbers: ”);
scanf(“%d%d”,&numbera, &numberb);

printf(“The two numbers entered are %d %d\n\n”,
numbera, numberb);
```

Ένα πρόγραμμα σε C

Πάντα υπάρχει μια και μόνο μια συνάρτηση `main`

```
#include <stdio.h>

int main (void ) {

  printf ("Hello world!\n");

  return 0;
}
```

Ο κώδικας οργανώνεται με άγκιστρα

Γράφουμε κώδικα μόνο μέσα σε συναρτήσεις!!!

Δεύτερο Πρόγραμμα σε C

```
#include <stdio.h>

void main (void) {
    int a, b, c, sum;

    scanf("%d", &a);
    scanf("%d", &b);
    scanf("%d", &c);

    sum = a + b + c;

    printf ("sum is %d", sum);
return;
}
```

τι κάνει;

Ένα πρόγραμμα σε C

```
#include <stdio.h>
int main(void) {
    int i, j;
    int sum ;
    i = 5 ;
    j = 6 ;
    sum = i + j ;
    printf ("sum: %d\n", sum);
    return 0;
}
```

Δηλώνουμε τις μεταβλητές πριν τα χρησιμοποιήσουμε !!!

ΣΥΝΑΡΤΗΣΕΙΣ

Παράδειγμα

- Γράψτε ένα πρόγραμμα που να τυπώνει το πιο κατω σχημα:

Παράδειγμα – Λύση 1

```
int main(){
printf("****\n");
printf("*   *\n");
printf("*   *\n");
printf("****\n");

printf("****\n");
printf("*   *\n");
printf("*   *\n");
printf("****\n");

printf("****\n");
printf("*   *\n");
printf("*   *\n");
printf("****\n");
return 0;
}
```

Ροή Ελέγχου

Εκτέλεση ίδιων εντολών σημαίνει ξαναγράψιμο ίδιων εντολών!

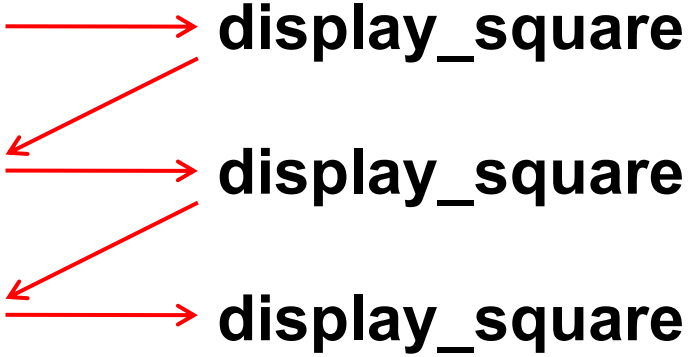
Παράδειγμα – Λύση 2

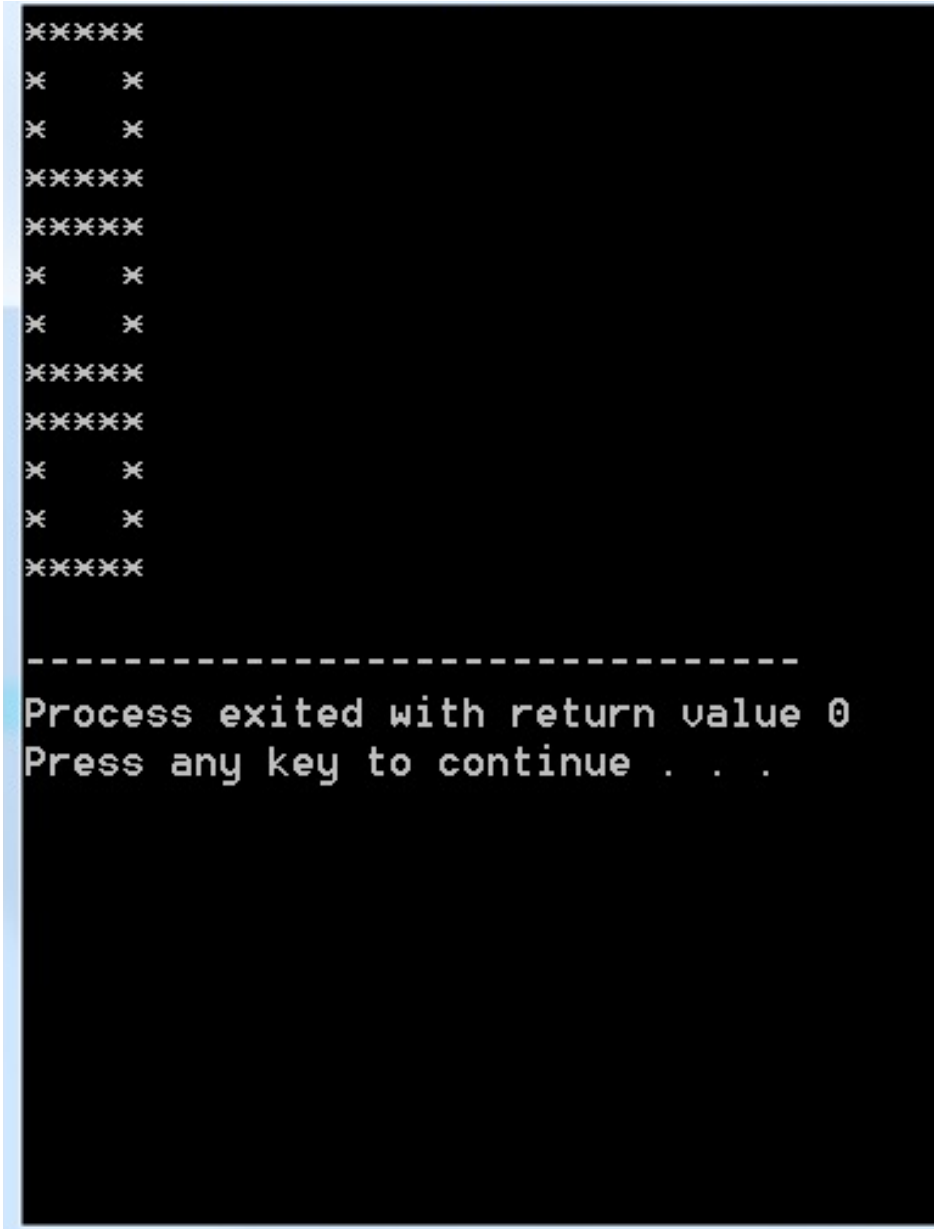
```
void display_square(){
printf("****\n");
printf("*   *\n");
printf("*   *\n");
printf("****\n");
 }

int main(){
display_square();

display_square();

display_square();

return 0;
}
```

Ίδιο πρόγραμμα με συνάρτηση

Επαναχρησιμοποίηση ίδιου κώδικα

Ροή Ελέγχου




*****
*   *
*   *
*****
*****
*   *
*   *
*****
*****
*   *
*   *
*****

Process exited with return value 0
Press any key to continue . . .

Παράδειγμα

τυπικοί παραμέτροι

πρωτότυπο συνάρτησης

ορισμός συνάρτησης

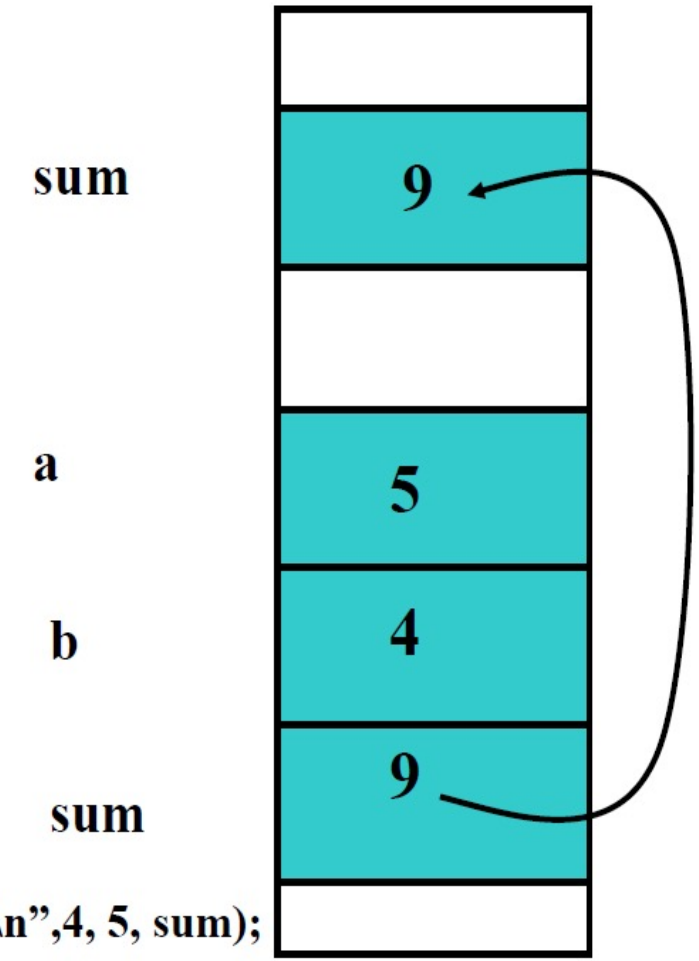
Ορίσματα/ πραγματικοί παραμέτροι

κλήση συνάρτησης

```
#include <stdio.h>
int compute_sum(int x, int y);

int compute_sum(int a, int b)
{
        int sum;
        sum = a + b;
        return sum;
}

int main()
{
        int sum=0;

        sum = compute_sum(5, 4);

        printf("The sum of %d and %d is %d\n",4, 5, sum);
        return(0);
}
```

Παράδειγμα – (συν.)

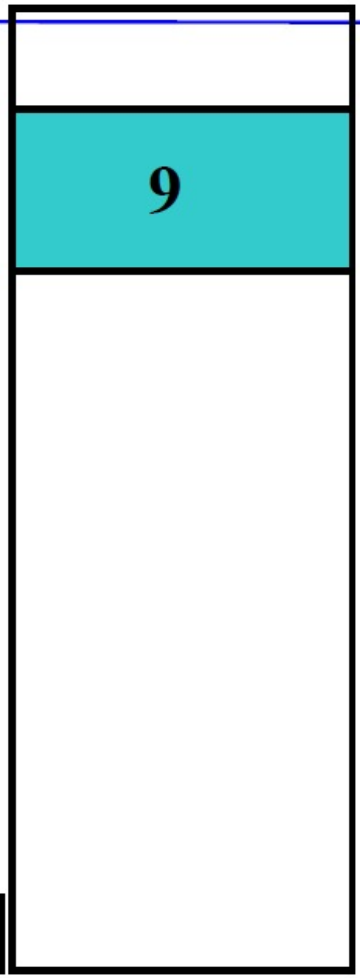
```
#include <stdio.h>
int compute_sum(int x, int y);

int compute_sum(int a, int b)
{
        int sum;
        sum = a + b;
        return sum;
}

int main()
{
        int sum=0;
        sum = compute_sum(5, 4);
        printf("The sum of %d and %d is %d\n",4, 5, sum);
        return(0);
}
```



Παράδειγμα – (συν.)

```
#include <stdio.h>
int compute_sum(int x, int y);

int compute_sum(int a, int b)
{
        int sum;
        sum = a + b;
        return sum;
}

int main()
{
        int sum=0;

        sum = compute_sum(5, 4);

        printf("The sum of %d and %d is %d\n",4, 5, sum);
        return(0);
}
```

sum



έκδοση 0: θα πρέπει να είναι εκτελέσιμο!

```c
#include <stdio.h>
int diabase();
int ypologisepower(int);
void typwse(int);

main( ) {
 int number, power;
 number = diabase( );
 power = ypologisepower(number);
 typwse(power);

}
```

```c
int diabase ( ) {
 printf ("function: diabase\n");
 return 5;
}

int ypologisepower(int x ) {
 printf ("function: ypologise\n");
 return x;
}

void typwse (int x ) {
  printf("function: typwse\n");
}
```

έκδοση 1: πλήρης typwse()

```c
#include <stdio.h>
int diabase();
int ypologisepower(int);
void typwse(int);

main( ) {
 int number, power;
 number = diabase( );
 power = ypologisepower(number);
 typwse(power);

}
```

```c
int diabase ( ) {
 printf ("function: diabase\n");
 return 5;
}

int ypologisepower(int x ) {
 printf ("function: ypologise\n");
 return x;
}

void typwse (int x ) {
 printf("function: typwse\n");
 printf("apotelesma: %d\n",x);
}
```

έκδοση 2: πλήρης ypologisepower()

```c
#include <stdio.h>
int diabase();
int ypologisepower(int);
void typwse(int);

main( ) {
 int number, power;
 number = diabase( );
 power = ypologisepower(number);
 typwse(power);

}
```

```c
int diabase ( ) {
 printf ("function: diabase\n");
 return 5;
}

int ypologisepower(int x ) {
 printf ("function: ypologise\n");
 return x * x * x;
}

void typwse (int x ) {
 printf("function: typwse\n");
 printf("apotelesma: %d\n",x);
}
```

Έκδοση 3: πλήρης diabase()

```c
#include <stdio.h>
int diabase();
int ypologisepower(int);
void typwse(int);

main( ) {
 int number, power;
 number = diabase( );
 power = ypologisepower(number);
 typwse(power);

}
```

```c
int diabase ( ) {
 int aninput;
 printf ("function: diabase\n");
 scanf("%d", &aninput);
 return aninput;
}

int ypologisepower(int x ) {
 printf ("function: ypologise\n");
 return x * x * x;
}

void typwse (int x ) {
 printf("function: typwse\n");
 printf("apotelesma: %d\n",x);
}
```

Ευχαριστώ για την προσοχή σας

Επικοινωνία

- Skype: fidas.christos
- Email: fidas@upatras.gr
- Phone: 2610 – 996491
- Web: http://cfidas.info

Άμεση Επικοινωνία μέσω Skype

SkypeID:
fidas.christos

- Ώρες γραφείου: Τετάρτη & Παρασκεύη 11:00-13:00

Join Zoom Meeting
https://upatras-gr.zoom.us/j/95080297961?pwd=MzRta0JRd3ZwVEVrREZNc09qbG1Zdz09

Το υλικό της διάλεξης είναι διαθέσιμο στο eclass

- https://eclass.upatras.gr/

ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Δρ. Χρήστος Α. Φείδας, Επίκουρος Καθηγητής

Ακαδημαϊκό Έτος 2021-2022

Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών
Πανεπιστήμιο Πατρών

1η Εβδομάδα: Εισαγωγικές Έννοιες

Τελεστής Ανάθεσης (assignment)

- Σύνταξη: **μεταβλητή = έκφραση;**
  - area = PI * radius * radius;
  - count = count + 1;
  - new_number = old_number;
  - average = total / count;

- Η τιμή της έκφρασης αποθηκεύεται στην μεταβλητή και ο τύπος της τιμής της έκφρασης μετατρέπεται στον τύπο της μεταβλητής

Αριθμητικοί τελεστές

Όνομα	Τελεστής	Παράδειγμα
Πρόσθεση	+	num1 + num2
Αφαίρεση	-	initial - spent
Πολλ/σμός	*	age * 6
Διαίρεση	/	sum / count
Υπόλοιπο	%	m % n

Τελεστές και προτεραιότητα

Τελεστές	Προτεραιότητα
()	Εάν είναι φωλιασμένο, τότε προτεραιότητα δίνεται στον πιο εσωτερικό. Εάν υπάρχουν πολλοί στο ίδιο επίπεδο τότε η προτεραιότητα είναι από αριστερά προς τα δεξιά (x+(y+z)*(x-y)*3)
+- (Μοναδιαίοι)	Εάν υπάρχουν πολλοί, η προτεραιότητα είναι από δεξιά προς αριστερά x=++y+--z;
* / %	Εάν υπάρχουν πολλοί, η προτεραιότητα είναι από αριστερά προς τα δεξιά 3*7*y/x
+ - (Δυαδικοί)	Εάν υπάρχουν πολλοί, η προτεραιότητα είναι από αριστερά προς τα δεξιά 3+x-y
=	Εάν υπάρχουν πολλοί η προτεραιότητα είναι από δεξιά προς αριστερά x=y=3

Τελεστές και Μετατροπή τύπων

- Αυτόματη μετατροπή
  - Σε ανάθεση, η τιμή στα δεξιά του = μετατρέπεται στον τύπο της μεταβλητής στα αριστερά του =
    - int x = 3.14; //στο x τελικά καταχωρείται η τιμή 3
    - float x = 2/3; //στο x καταχωρείται η τιμή 0.0000
- Ρητή μετατροπή (Casting),
- float x1,x2,x3,x4;
  - x1 = 2/3; x1=0.0000
  - x2 = (float)2/3; x2=0.6666
  - x3 = 2/(float)3; x3=0.6666
  - x4 = (float)(2/3); x4=0.0000

Σχεσιακοί Τελεστές (Relational Operators)

- Δυαδικοί Τελεστές
  - **<** μικρότερο από
  - **>** μεγαλύτερο από
  - **<=** μικρότερο ή ίσο με
  - **>=** μεγαλύτερο ή ίσο με
  - **==** ίσο με
  - **!=** διάφορο του
- Αποτιμούνται σε 0 (ψευδές) ή 1 (αληθές)
- Τύποι τελεστέων int, char, float, double

Λογικοί Τελεστές (Logical Operators)

- Συνδυάζουν δύο λογικές παραστάσεις σε μια σύνθετη λογική παράσταση
  - **&&** σύζευξη, δυαδικός τελεστής (**AND**)
  - **||** διάζευξη, δυαδικός τελεστής (**OR**)
  - **!** άρνηση, μοναδιαίος τελεστής (**NOT**)
- Αποτιμούνται σε 0 ή 1
  - **0** (δεν ισχύει, ψευδές ή **false**)
  - **1** (ισχύει, αληθές ή **true**)

Τελεστές postfix/prefix

προ-σημειογραφική (prefix)

++i;	<==>	i = i + 1;
--i;	<==>	i = i - 1;

μετα-σημειογραφική (postfix)

i++;	<==>	i = i + 1;
i--;	<==>	i = i - 1;

Prefix vs Postfix

```
i = 5;
x = ++i;
y = i++;
```

- Το **x είναι 6**, το **y είναι 6** και το **i είναι 7**
- Χρησιμοποιείτε -- ++ σε απλές εκφράσεις
  - `n = ++m;` /*first increment m, then assign its value to n*/
  - `n = m++;` /*first assign m's value to n, then increment m*/

Σύνθετοι Τελεστές Ανάθεσης

- `i+=k;` `<==> i = i + k;`
- `i*=k;` `<==> i = i * k;`
- `i`***`operator`*** `= k; <==> i = i` ***`op`*** `(k);`

Δομές Εκτέλεσης

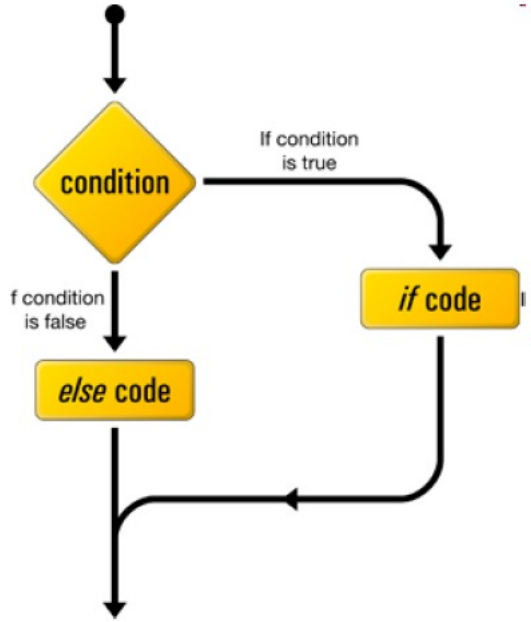
- **Διαδοχική εκτέλεση**
  - Στηρίζεται στην απλή παράθεση εκφράσεων/εντολών, η μια μετά την άλλη
- **Εκτέλεση με επιλογή**
  - Η ροή του προγράμματος “διακόπτεται” για να παρθεί μια απόφαση, να γίνει κάποια επιλογή
  - Το αποτέλεσμα της απόφασης καθορίζει την “κατεύθυνση” της ροής του προγράμματος
- ***Εκτέλεση με επανάληψη***
  - Μια ομάδα εκφράσεων/εντολών εκτελείται περισσότερο από μια φορά

ΔΟΜΕΣ ΕΛΕΓΧΟΥ

Εντολή διακλάδωσης if-else

Συντάσσεται ως εξής:

```
if (condition)
{
      ... // block A
}
else
{
      ... // block B
}
```



Εάν η συνθήκη (condition) είναι αληθής, εκτελείται το block των εντολών που περικλείεται μεταξύ των πρώτων {} (block A).

Αλλιώς, εκτελείται το block των εντολών μετά το **else** (block B).

Παράδειγμα 1: if-else – Λύση

```
int main(void){
    int a, b;

    printf("dwse ta a kai to b\n");
    scanf("%d%d", &a, &b);

    if (a < b)
          printf("%d\n",  a);
    else
          printf("%d\n",  b);

return 0;
}
```

Ενναλακτικές Λύσεις

```
int min(int a, int b){
    int minimum;
    if (a < b)
        minimum = a;
    else
        minimum = b;
    return minimum;
}
```

```
int min(int a, int b)
{
    if (a < b)
        return a;
    else
        return b;
}
```

```
int min(int a, int b){
    if (a < b)
        return a;
    return b;
}
```

```
int min(int a, int b){
    int minimum;
    minimum = b;

    if (a < b)
        minimum = a;

    return minimum;
}
```

Παράδειγμα if-else

- Γράψτε τη συνάρτηση min_three() η οποία παίρνει σαν παράμετρους τρεις ακέραιους αριθμούς και επιστρέφει τον μικρότερο εξ αυτών

Παράδειγμα if-else - Λύση

```
int min_three(int a, int b, int c){
    int minimum;
    minimum = min(a,b);
    minimum = min(minimum,c);

    return minimum;
}
```

Με χρήση της συνάρτησης **min()** την οποία υλοποιήσαμε στο Παράδειγμα 2

Ενναλακτική Λύση

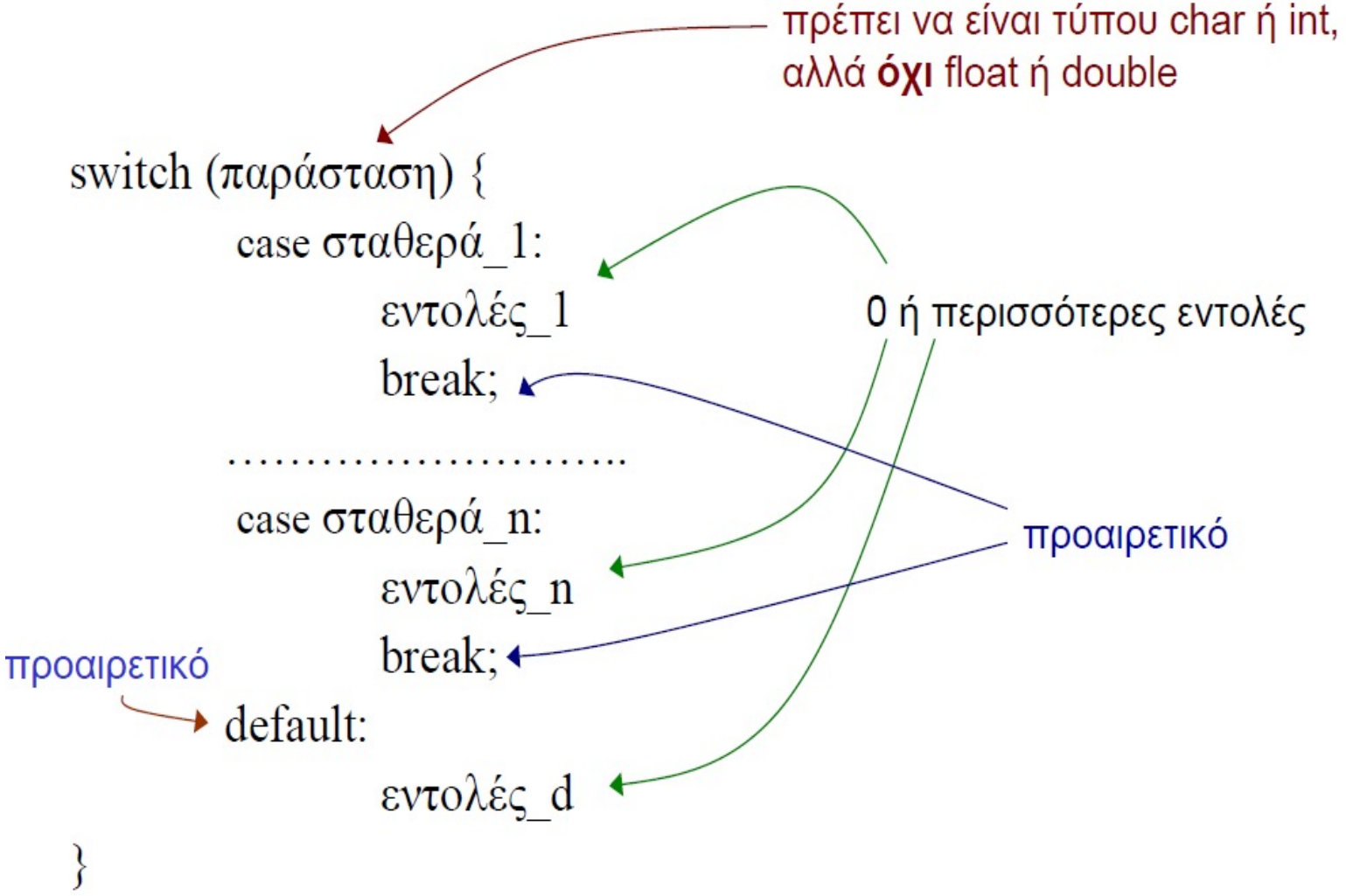
```
int min_three(int a, int b, int c){
        int minimum;
        if (a < b)
                minimum = a;
        else
                minimum = b;
        if (c < minimum)
                minimum = c;
        return minimum;
}
```

Χωρίς τη χρήση οποιασδήποτε συνάρτησης

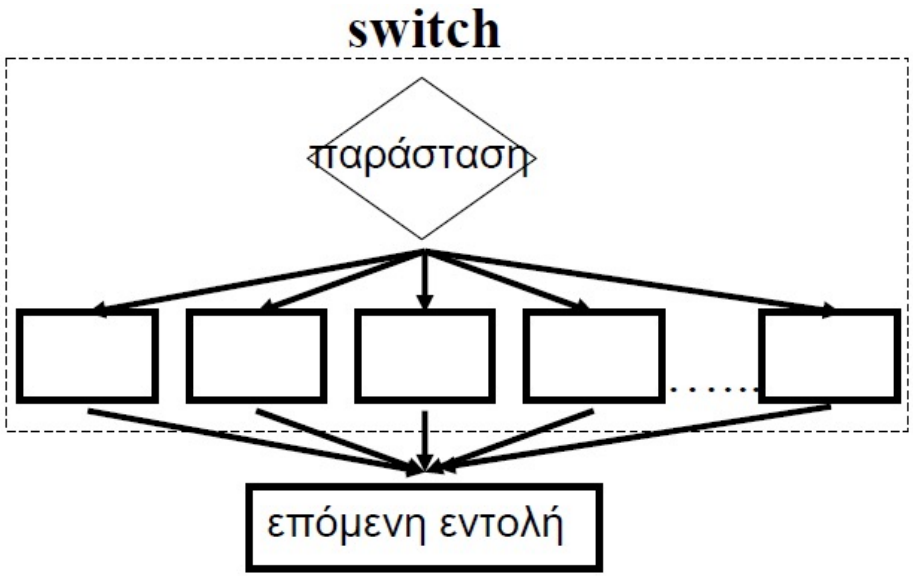
Φώλιασμα (nesting)

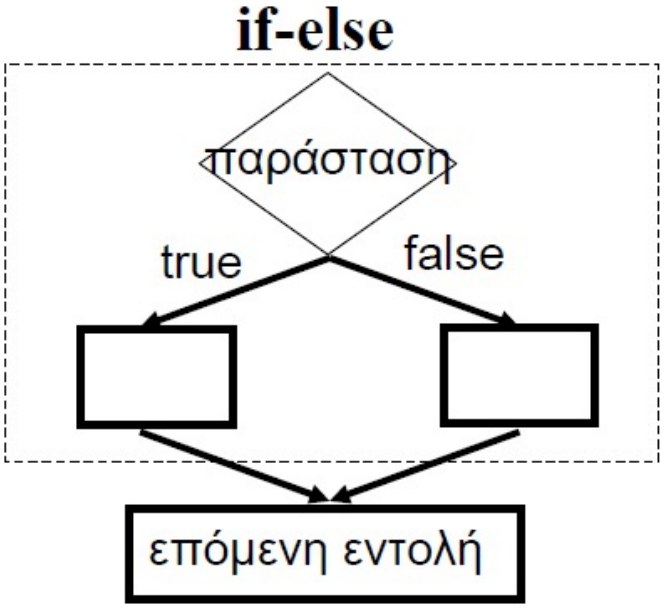
```
int min_three(int a, int b, int c){
        int minimum;
        if (a < b)
            if (a < c)
                minimum = a;
            else
                minimum = c;
        else
            if (b < c)
                minimum = b;
            else
                minimum = c;

        return minimum;
}
```

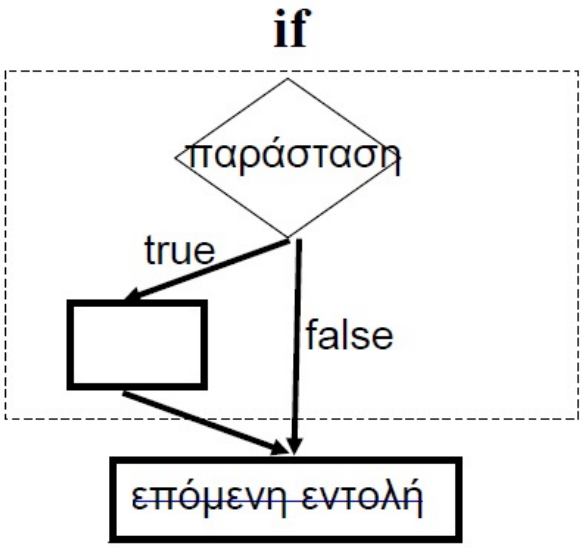
Εντολή switch



Switch, if-else, if







ΔΟΜΕΣ ΕΠΑΝΑΛΗΨΗΣ

Εντολή while

- Η εντολή ***while*** επαναλαμβάνει την εκτέλεση μιας πρότασης ενώσω η τιμή μιας λογικής παράστασης είναι αληθής (1)
- Η σύνταξη της εντολής είναι:

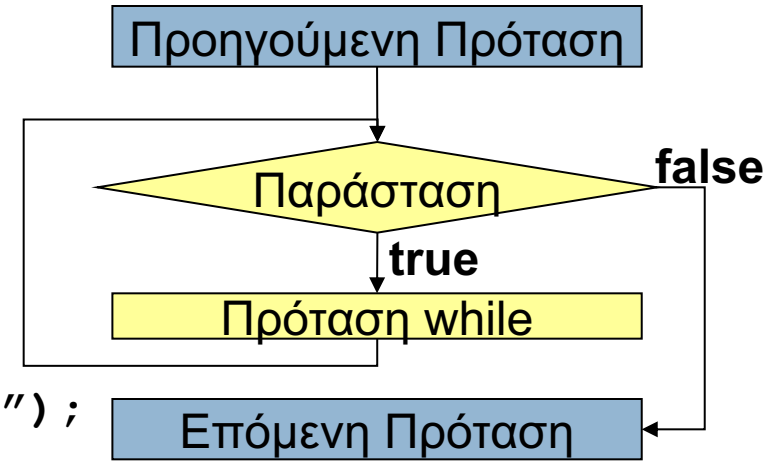
```
while (παράσταση)
   Πρόταση1;
```

```
while (παράσταση) Πρόταση1;
```

```
while (παράσταση)
{
   Πρόταση1_1;
   Πρόταση1_2;
   …
   Πρόταση1_N;
}
```

- Η *πρόταση1* (απλή ή σύνθετη) θα εκτελείται συνέχεια μέχρι η *παράσταση* της ***while*** να πάρει λογική τιμή false (0)
- Π.χ.

```
while (5>0)
{
   printf("Δε θα σταματήσω ποτέ\n");
}
```



Σύνταξη while

```
while   (έκφραση)
    εντολή;
```

ή

```
while   (έκφραση) {
    εντολή;
    εντολή;
    …
}
```

Σημασία while

- Όσον η τιμή της έκφρασης (συνθήκης) είναι αληθής – δηλαδή διάφορη του μηδέν – εκτέλεσε τις εξαρτώμενες εντολές, *αλλιώς συνέχισε με τις εντολές που ακουλουθούν (μετά) το* while *block*

while με μετρητή

```
int x;
x=0;
while   (x   <   5)
{
    printf("%d\n",x);
    x = x + 1;
}
```

μεταβλητή που χρησιμοποιείται για έλεγχο επανάληψης (control/ induction variable)

αρχικοποίηση

συνθήκη επανάληψης

επόμενο βήμα

while (συν.)

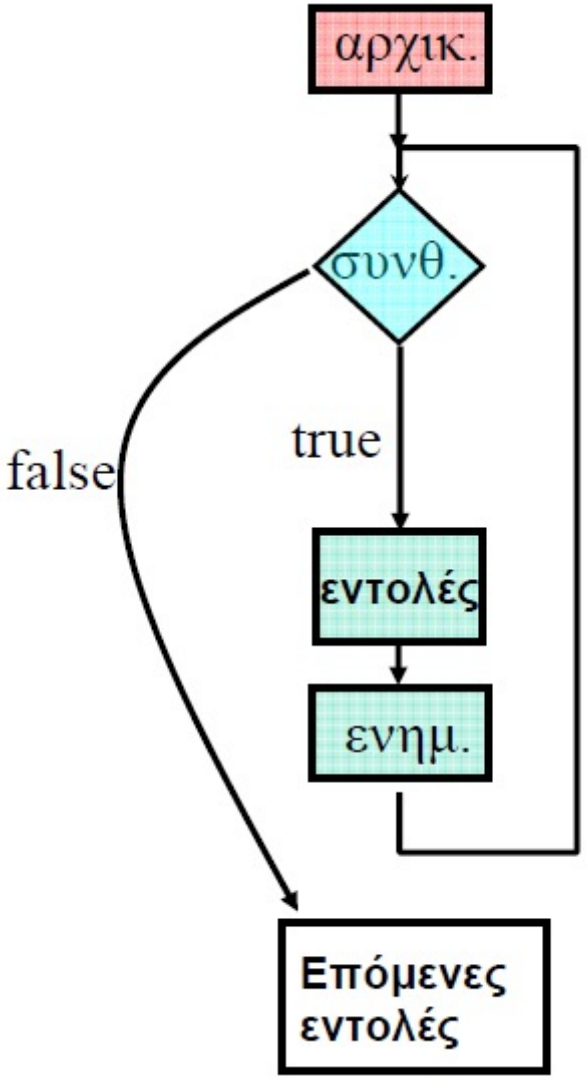
x	x<5	output
0	1	0
1	1	1
2	1	2
3	1	3
4	1	4
5	0	

Άθροισμα σειράς – Κώδικας

```
int number;        /* holds input number one at a time */
int sum;                                  /* current sum */

scanf("%d",&number);          /* diabase prwto stoixeio */
sum = 0;                                 /* arxikopoihsh */
while(number != 0){
     sum = sum + number;                  /* epeksergasia*/
     scanf("%d",&number); /* diavase epomeno stoixeio*/
}
printf("To athroisma tis seiras einai %d\n",sum); /*eksodos*/
```

Ροή Ελέγχου for

```
int  x,y;

for  (x=0;  x<5;  ++x){
        printf("%d\n",x);
}

y = x*x + 10;
printf("%d\n",y);
```



Ομοιότητα for με while

```
int x;
x=0;
while (x < 5) {
    printf("%d\n",x);
    ++x;
}
```

```
int x;
for (x=0; x<5;++x) {
    printf("%d\n",x);

}
```

Οποιοδήποτε for μπορεί να γραφεί με while και οποιοδήποτε while με for

Σύνταξη do-while

```
do
    εντολή;
while(συνθήκη);
```

```
do{
    εντολή;
    εντολή;
}while(συνθήκη);
```

Σημασία do-while

- Εκτέλεσε το σώμα της δομής do-while
- **Εφόσον η συνθήκη ισχύει** επανέλαβε την εκτέλεση των εντολών στο σώμα του βρόχου

- Στα for και while το σώμα μπορεί να μην εκτελεστεί εάν η συνθήκη δεν ικανοποιείται, ενώ στο do-while εκτελείται *τουλάχιστον μια φορά!*

Εντολή **break** (πρόωρη έξοδος)

- Η εντολή break μας **βγάζει από την επανάληψη**

```
while(count<n){
        scanf("%d", &number);
        if (number == -1)
            break;
        ++count;
}
printf("%d  %d\n",  count,  n);
```

Εντολή **continue**

Παράλειψη των υπολοίπων εντολών σε μια εντολή επανάληψης και εκ νέου εκτέλεση του σώματος εντολών

```
int count = 0, n_students = 27;
float number, sum = 0.0;
while(count < n_students){
      scanf("%d",&number);
      if (number < 0 || number > 100){
         printf("wrong entry\n");
         continue;
      }
      sum += number;
      ++count;
}
printf("The average is %f\n", sum/n_students);
```

Ευχαριστώ για την προσοχή σας

Επικοινωνία

- Skype: fidas.christos
- Email: fidas@upatras.gr
- Phone: 2610 – 996491
- Web: http://cfidas.info

Άμεση Επικοινωνία μέσω Skype

SkypeID:
fidas.christos

- Ώρες γραφείου: Τετάρτη & Παρασκεύη 11:00-13:00

Join Zoom Meeting
https://upatras-gr.zoom.us/j/95080297961?pwd=MzRta0JRd3ZwVEVrREZNc09qbG1Zdz09

Το υλικό της διάλεξης είναι διαθέσιμο στο eclass

- https://eclass.upatras.gr/