

Διάλεξη 10 - Πίνακες

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Εισαγωγή στον Προγραμματισμό

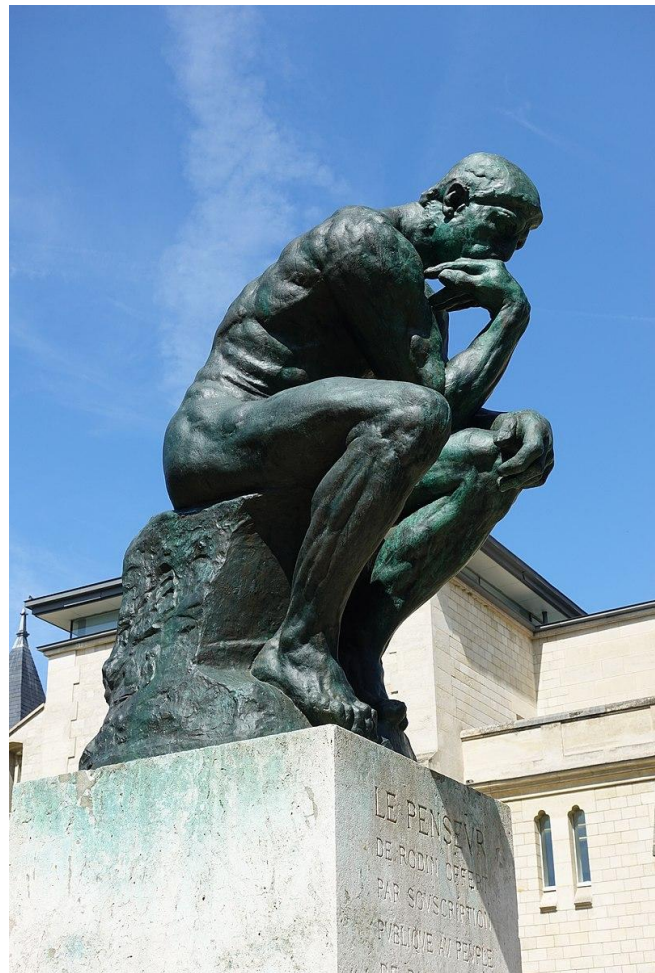
Θανάσης Αυγερινός

Ανακοινώσεις / Διευκρινήσεις

- Το Linux εργαστήριο είναι διαθέσιμο κατά την διάρκεια της μέρας
- Υπάρχει ενδιαφέρον για πρόσθετο εργαστήριο με βοηθούς του μαθήματος;
 - a. Παρασκευή ή άλλη μέρα;
- Υπενθύμιση: Η προθεσμία της εργασίας είναι την Δευτέρα 23:59
 - a. Μην δεχτώ μηνύματα Τρίτη 00:01: δεν πέρασε το commit, τι κάνω;
 - b. Και το GitHub έχει outages, μην κάνετε commit τελευταίο λεπτό!

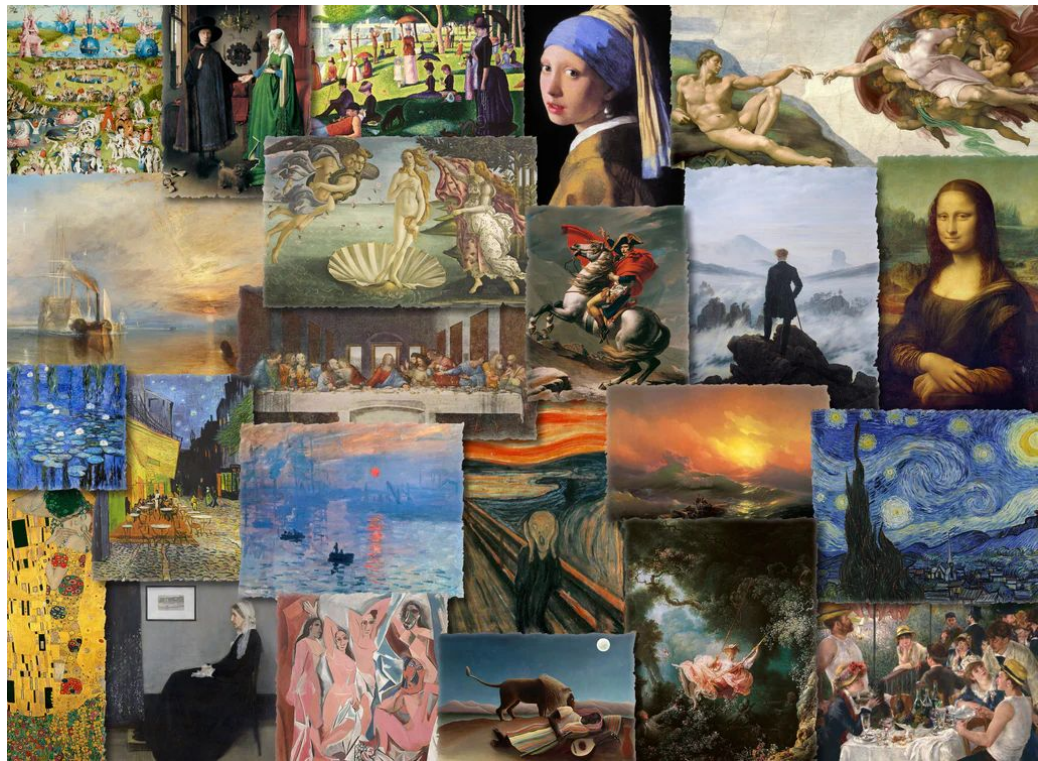
Την προηγούμενη φορά

- Δεδομένα Εισόδου και εξόδου
- Παραδείγματα



Σήμερα

- Λίγο για scanf
- Πίνακες (Arrays)



Θέλω να διαβάσω δύο αριθμούς από την πρότυπη είσοδο και να τους προσθέσω. Πως;

```
$ ./addnums
```

```
Give me a number: 40
```

```
Give me another number: 2
```

```
Total: 42
```

Κάνοντας χρήση της `getchar` και φτιάχνοντας μια συνάρτηση `getinteger` προκειμένου να διαβάσουμε τους χαρακτήρες έναν-έναν και να τους μετατρέψουμε σε αριθμό. Υπάρχει άλλος τρόπος να επιτύχουμε το ίδιο αποτέλεσμα;

```
#define ERROR -1

int getinteger(int base) {
    int ch;
    int val = 0;
    while ((ch = getchar()) != '\n')
        if (ch >= '0' && ch <= '0' + base - 1)
            val = base * val + (ch - '0');
        else
            return ERROR;
    return val;
}
```

Η συνάρτηση `scanf`

Η συνάρτηση `scanf` ορίζεται στο header file `stdio.h` και χρησιμοποιείται για να διαβάσει δεδομένα εισόδου πολλών τύπων από το `stdin` του προγράμματος και να αποθηκεύσει τις τιμές τους σε μεταβλητές. Αν επιτύχει, επιστρέφει πόσα δεδομένα εισόδου διάβασε. Αν αποτύχει, επιστρέφει την τιμή End-Of-File / EOF (-1).

Πως μπορώ να βρω πως συμπεριφέρεται;

Ανοίγω ένα τερματικό και τρέχω
`man scanf!`

Η συνάρτηση scanf

Η συνάρτηση **scanf** ορίζεται στο header file `stdio.h` και χρησιμοποιείται για να διαβάσει δεδομένα εισόδου πολλών τύπων από το **stdin** του προγράμματος και να αποθηκεύσει τις τιμές τους σε μεταβλητές. Αν επιτύχει, επιστρέφει *πόσα δεδομένα εισόδου* διάβασε. Αν αποτύχει, επιστρέφει την τιμή **End-Of-File / EOF** (-1). Η συνάρτηση έχει την ακόλουθη μορφή:

```
int scanf(const char *restrict format, ...);
```

Έχει μια συμβολοσειρά μορφοποίησης
(format string)

Δέχεται όσα ορίσματα περάσουμε
(άλλο μάθημα)

Η συνάρτηση scanf

Η συνάρτηση **scanf** ορίζεται στο header file `stdio.h` και χρησιμοποιείται για να διαβάσει δεδομένα εισόδου πολλών τύπων από το **stdin** του προγράμματος και να αποθηκεύσει τις τιμές τους σε μεταβλητές. Αν επιτύχει, επιστρέφει πόσα δεδομένα εισόδου διάβασε. Αν αποτύχει, επιστρέφει την τιμή End-Of-File / EOF (-1). Η συνάρτηση έχει την ακόλουθη μορφή:

```
int scanf(const char *restrict format, ...);
```

```
int printf(const char *restrict format, ...);
```

Είναι η συμμετρική της printf για διάβασμα αντί για εκτύπωση

Χρήση της συνάρτησης scanf

Τι κάνει το παρακάτω πρόγραμμα;

```
#include <stdio.h>

int main() {
    int n;
    printf("Gimme a number: ");
    scanf("%d", &n);
    printf("Square: %d\n", n * n);
    return 0;
}
```

Χρήση της συνάρτησης scanf

Τι κάνει το παρακάτω πρόγραμμα;

```
#include <stdio.h>

int main() {
    int n;
    printf("Gimme a number: ");
    scanf("%d", &n);
    printf("Square: %d\n", n * n);
    return 0;
}
```

Περνάμε την **διεύθυνση (spoiler)** της μεταβλητής `n` στην μνήμη ώστε η `scanf` να μπορέσει να αναθέσει την τιμή που διάβασε

Τυπώνει στο `stdout` το τετράγωνο του αριθμού που γράψαμε στο `stdin`

Χρήση της συνάρτησης scanf

Τι κάνει το παρακάτω πρόγραμμα;

```
#include <stdio.h>

int main() {

    int n;

    printf("Gimme a number: ");
    scanf("%d", &n);
    printf("Square: %d\n", n * n);
    return 0;
}
```

Τι θα γινόταν αν γράφαμε `scanf("%d", n);`; Γιατί;
\$./scanf
Gimme a number: 3
Segmentation fault

Περνάμε την **διεύθυνση (spoiler)** της μεταβλητής `n` στην μνήμη ώστε η `scanf` να μπορέσει να αναθέσει την τιμή που διάβασε

Τυπώνει στο `stdout` το τετράγωνο του αριθμού που γράψαμε στο `stdin`

Χρήση της συνάρτησης scanf

Τι κάνει το παρακάτω πρόγραμμα;

```
#include <stdio.h>

int main() {
    int n;
    printf("Gimme a number: ");
    scanf("%d", &n);
    printf("Square: %d\n", n * n);
    return 0;
}
```

Είναι σωστό αυτό το πρόγραμμα;

Χρήση της συνάρτησης scanf

Τι κάνει το παρακάτω πρόγραμμα;

```
#include <stdio.h>

int main() {
    int n;
    printf("Gimme a number: ");
    scanf("%d", &n);
    printf("Square: %d\n", n * n);
    return 0;
}
```

Είναι σωστό αυτό το πρόγραμμα;

Όχι καθώς δεν ελέγχουμε την τιμή επιστροφής της scanf (EOF ή ίσως 0!)

Χρήση της συνάρτησης scanf

Τι κάνει το παρακάτω πρόγραμμα;

```
#include <stdio.h>

int main() {
    int n;

    printf("Gimme a number: ");
    scanf("%d", &n);
    printf("Square: %d\n", n * n);
    return 0;
}
```

```
$ ./scanf
Gimme a number: 16
Square: 256
$ ./scanf
Gimme a number: Square:
1068701481
$ ./scanf
Gimme a number: hello
Square: 1072038564
```

Χρήση της συνάρτησης `scanf` - Πολλά ορίσματα

Τι κάνει το παρακάτω πρόγραμμα;

```
#include <stdio.h>

int main() {
    int n1, n2;
    printf("Gimme two numbers: ");
    scanf("%d %d", &n1, &n2);
    printf("Result: %d\n", n1 * n2);
    return 0;
}
```

Χρήση της συνάρτησης scanf - Πολλά ορίσματα

Τι κάνει το παρακάτω πρόγραμμα;

```
#include <stdio.h>

int main() {
    int n1, n2;
    printf("Gimme two numbers: ");
    scanf("%d %d", &n1, &n2);
    printf("Result: %d\n", n1 * n2);
    return 0;
}
```

```
$ ./scanf2
Gimme two numbers: 2 4
Result: 8
```

Καθώς ψάχνει για δεκαδικό ψηφίο, η scanf αγνοεί τους κενούς χαρακτήρες ή αλλαγές γραμμής

Τι κάνει το παρακάτω πρόγραμμα;

```
#define ERROR -1                                // Return value for illegal character

int getinteger(int base) {

    int ch;                                     // No need to declare ch as int - no EOF handling

    int val = 0;                                // Initialize return value

    while ((ch = getchar()) != '\n')           // Read up to new line

        if (ch >= '0' && ch <= '0' + base - 1) // Legal character?

            val = base * val + (ch - '0');      // Update return value

        else

            return ERROR;    // Illegal character read

    return val;    // Everything OK - Return value of number read

}
```

Τι κάνει το παρακάτω πρόγραμμα;

```
int i, ch, total = 0;

int letfr[26]; // Letter occurrences and frequencies array

for (i=0 ; i < 26 ; i++)

    letfr[i] = 0;

while ((ch = getchar()) != EOF) {

    if (ch >= 'A' && ch <= 'Z') {

        letfr[ch-'A']++;           // Found upper case letter

        total++;

    }

    if (ch >= 'a' && ch <= 'z') {

        letfr[ch-'a']++;           // Found lower case letter

        total++;

    }

}
```

Πίνακες

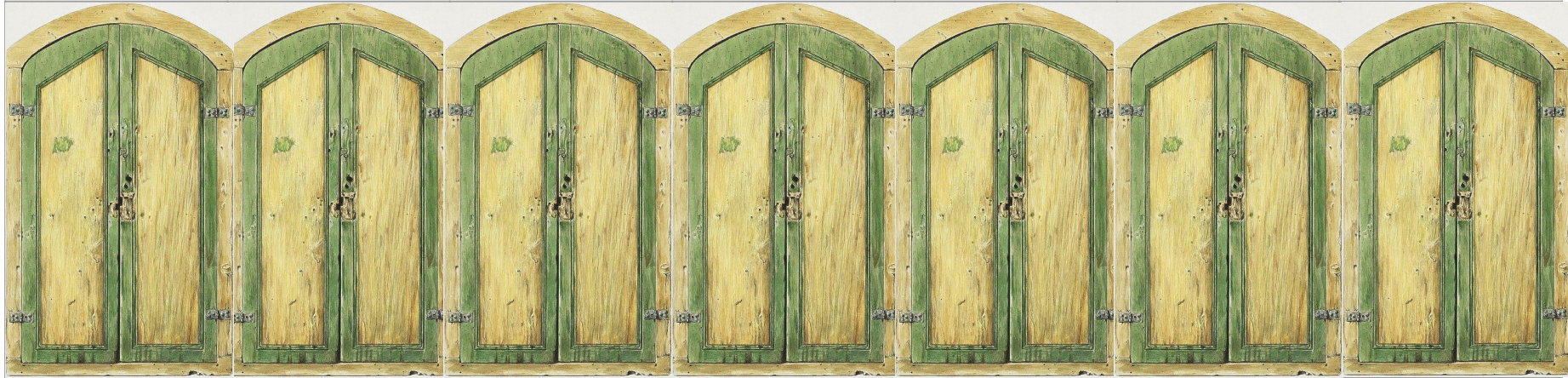
Ένας Γρίφος

Έστω ότι έχω 100 αρκουδάκια και
ψάχνω να βρω το μεγαλύτερο. Τι
μπορώ να κάνω για να το βρω;





Στόχος: ελαχιστοποίηση του κόπου (efficient aka τεμπέλης)



Πως θα κωδικοποιήσουμε αυτό το πρόβλημα σε C;

Έστω ότι χρησιμοποιούμε έναν ακέραιο (int) για να αναπαραστήσουμε το κάθε αρκουδάκι. Ας γράψουμε τον κώδικα:

Πως θα κωδικοποιήσουμε αυτό το πρόβλημα σε C;

Έστω ότι χρησιμοποιούμε έναν ακέραιο (int) για να αναπαραστήσουμε το κάθε αρκουδάκι. Ας γράψουμε τον κώδικα:

```
int bear0, bear1, bear2, bear3, bear4, bear5, bear6, bear7, bear8, bear9, bear10, bear11, bear12, bear13,
bear14, bear15, bear16, bear17, bear18, bear19, bear20, bear21, bear22, bear23, bear24, bear25, bear26,
bear27, bear28, bear29, bear30, bear31, bear32, bear33, bear34, bear35, bear36, bear37, bear38, bear39,
bear40, bear41, bear42, bear43, bear44, bear45, bear46, bear47, bear48, bear49, bear50, bear51, bear52,
bear53, bear54, bear55, bear56, bear57, bear58, bear59, bear60, bear61, bear62, bear63, bear64, bear65,
bear66, bear67, bear68, bear69, bear70, bear71, bear72, bear73, bear74, bear75, bear76, bear77, bear78,
bear79, bear80, bear81, bear82, bear83, bear84, bear85, bear86, bear87, bear88, bear89, bear90, bear91,
bear92, bear93, bear94, bear95, bear96, bear97, bear98, bear99;
```

Πως θα κωδικοποιήσουμε αυτό το πρόβλημα σε C;

Έστω ότι χρησιμοποιούμε έναν ακέραιο (int) για να αναπαραστήσουμε το κάθε αρκουδάκι. Ας γράψουμε τον κώδικα:




















```
int bear0, bear1, bear2, bear3, bear4, bear5, ..., bear99, max;
```

```
bear0 = 42; bear1 = 4; bear2 = 2; ... // αρχικοποίηση μεταβλητών
```

```
// find the max here:
```

HOW LONG CAN YOU WORK ON MAKING A ROUTINE TASK MORE
EFFICIENT BEFORE YOU'RE SPENDING MORE TIME THAN YOU SAVE?
(ACROSS FIVE YEARS)

Πολύ επαναληπτικό -
μπορούμε να γλυτώσουμε
χρόνο;

		HOW OFTEN YOU DO THE TASK					
		50/DAY	5/DAY	DAILY	WEEKLY	MONTHLY	YEARLY
HOW MUCH TIME YOU SHAVE OFF	1 SECOND	 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
	5 SECONDS	 DAYS	12 HOURS	2 HOURS	21 MINUTES	5 MINUTES	25 SECONDS
	30 SECONDS	 4 WEEKS	 3 DAYS	12 HOURS	2 HOURS	30 MINUTES	2 MINUTES
	1 MINUTE	 8 WEEKS	 6 DAYS	 1 DAY	4 HOURS	1 HOUR	5 MINUTES
	5 MINUTES	9 MONTHS	 4 WEEKS	 6 DAYS	21 HOURS	5 HOURS	25 MINUTES
	30 MINUTES		6 MONTHS	 5 WEEKS	 5 DAYS	 1 DAY	2 HOURS
	1 HOUR		10 MONTHS	2 MONTHS	 10 DAYS	 2 DAYS	5 HOURS
	6 HOURS				2 MONTHS	 2 WEEKS	 1 DAY
	 1 DAY					 8 WEEKS	 5 DAYS

Δήλωση Πίνακα (Array) στην Γλώσσα C

Ένας **πίνακας** μας επιτρέπει να χειριζόμαστε ένα σύνολο από δεδομένα ίδιου τύπου με ενιαίο και γενικό τρόπο. Στην C η δήλωση ενός πίνακα έχει την μορφή:

τύπος όνομα[μέγεθος];

Ο **τύπος (type)** της μεταβλητής λέει στον μεταγλωττιστή πόση μνήμη να δεσμεύσει για κάθε στοιχείο του πίνακα

Το **όνομα (name)** του πίνακα κάνει τον μεταγλωττιστή να διαλέξει την διεύθυνση της μνήμης θα τον αποθηκεύσει

Το **μέγεθος (size)** του πίνακα λέει στον μεταγλωττιστή πόσες θέσεις αυτού του τύπου να κρατήσει - στατικό: αφού δηλωθεί δεν αλλάζει κατά την εκτέλεση

Δήλωση Πίνακα (Array) στην Γλώσσα C

Ένας **πίνακας** μας επιτρέπει να χειριζόμαστε ένα σύνολο από δεδομένα ίδιου τύπου με ενιαίο και γενικό τρόπο. Στην C η δήλωση ενός πίνακα έχει την μορφή:

```
int bears[100];
```

Ο **τύπος (type)** της μεταβλητής λέει στον μεταγλωττιστή πόση μνήμη να δεσμεύσει για κάθε στοιχείο του πίνακα

Το **όνομα (name)** του πίνακα κάνει τον μεταγλωττιστή να διαλέξει την διεύθυνση της μνήμης θα τον αποθηκεύσει

Το **μέγεθος (size)** του πίνακα λέει στον μεταγλωττιστή πόσες θέσεις αυτού του τύπου να κρατήσει - στατικό: αφού δηλωθεί δεν αλλάζει κατά την εκτέλεση

Δήλωση Πίνακα (Array) στην Γλώσσα C

Ένας **πίνακας** μας επιτρέπει να χειριζόμαστε ένα σύνολο από δεδομένα ίδιου τύπου με ενιαίο και γενικό τρόπο. Στην C η δήλωση ενός πίνακα έχει την μορφή:

```
int bears[100];
```

Μπορούμε να αναφερθούμε στο κάθε στοιχείο του πίνακα χρησιμοποιώντας την **θέση (index)** του στοιχείου στον πίνακα:
`bears[0], bears[1], ..., bears[99]`

	Μνήμη			
Bytes 0-3	b0	b1	b2	b3
Bytes 4-7	b4	b5	b6	b7
Bytes 8-11	b8	b9	b10	b11
Bytes 12-15	b12	b13	b14	b15
	...			
Bytes 400-403	b400	b401	b402	b403

Δήλωση Πίνακα (Array) στην Γλώσσα C

Ένας **πίνακας** μας επιτρέπει να χειριζόμαστε ένα σύνολο από δεδομένα ίδιου τύπου με ενιαίο και γενικό τρόπο. Στην C η δήλωση ενός πίνακα έχει την μορφή:

```
int bears[100];
```

Μπορούμε να αναφερθούμε στο κάθε στοιχείο του πίνακα χρησιμοποιώντας την **θέση (index)** του στοιχείου στον πίνακα:
`bears[0], bears[1], ..., bears[99]`

	Μνήμη			
bears[0] Bytes 0-3	b0	b1	b2	b3
Bytes 4-7	b4	b5	b6	b7
Bytes 8-11	b8	b9	b10	b11
Bytes 12-15	b12	b13	b14	b15
	...			
Bytes 400-403	b400	b401	b402	b403

Δήλωση Πίνακα (Array) στην Γλώσσα C

Ένας **πίνακας** μας επιτρέπει να χειριζόμαστε ένα σύνολο από δεδομένα ίδιου τύπου με ενιαίο και γενικό τρόπο. Στην C η δήλωση ενός πίνακα έχει την μορφή:

```
int bears[100];
```

Μπορούμε να αναφερθούμε στο κάθε στοιχείο του πίνακα χρησιμοποιώντας την **θέση (index)** του στοιχείου στον πίνακα:
`bears[0], bears[1], ..., bears[99]`

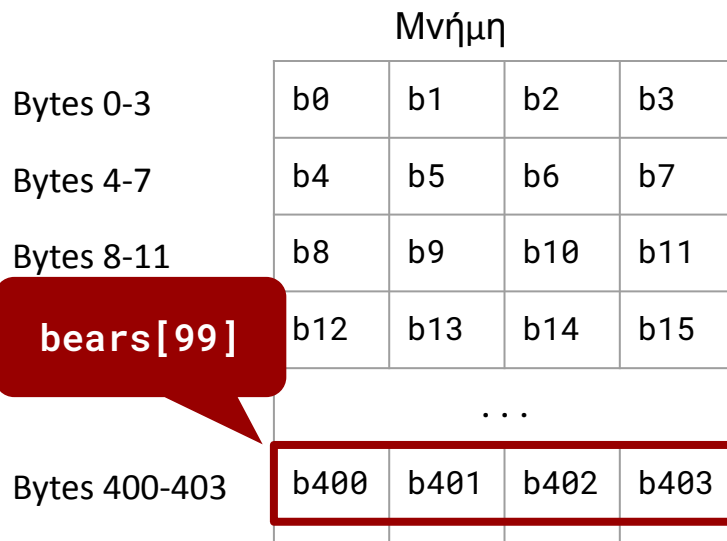
	Μνήμη			
	b0	b1	b2	b3
Bytes 4-7	b4	b5	b6	b7
Bytes 8-11	b8	b9	b10	b11
Bytes 12-15	b12	b13	b14	b15
	...			
Bytes 400-403	b400	b401	b402	b403

Δήλωση Πίνακα (Array) στην Γλώσσα C

Ένας **πίνακας** μας επιτρέπει να χειριζόμαστε ένα σύνολο από δεδομένα ίδιου τύπου με ενιαίο και γενικό τρόπο. Στην C η δήλωση ενός πίνακα έχει την μορφή:

```
int bears[100];
```

Μπορούμε να αναφερθούμε στο κάθε στοιχείο του πίνακα χρησιμοποιώντας την **θέση (index)** του στοιχείου στον πίνακα:
`bears[0], bears[1], ..., bears[99]`



Δήλωση Πίνακα (Array) στην Γλώσσα C

Ένας **πίνακας** μας επιτρέπει να χειριζόμαστε ένα σύνολο από δεδομένα ίδιου τύπου με ενιαίο και γενικό τρόπο. Στην C η δήλωση έχει την ακόλουθη μορφή:

```
int bears[100];
```

Ο πίνακας **bears**
καταλαμβάνει $4 * 100 = 400$ bytes
μνήμης στις
διευθύνσεις 4-403

Μπορούμε να αναφερθούμε στο κάθε στοιχείο του πίνακα χρησιμοποιώντας την **θέση (index)** του στοιχείου στον πίνακα:
`bears[0], bears[1], ..., bears[99]`

Μνήμη	b0	b1	b2	b3
Bytes 0-3				
Bytes 4-7	b4	b5	b6	b7
Bytes 8-11	b8	b9	b10	b11
Bytes 12-15	b12	b13	b14	b15
...	...			
Bytes 400-403	b400	b401	b402	b403

Χρήση Στοιχείων Πίνακα

Ένας πίνακας N στοιχείων, έχει στοιχεία με θέσεις από το 0 μέχρι το N-1

Κάθε στοιχείο του πίνακα μπορεί να χρησιμοποιηθεί όπως μια μεταβλητή του ίδιου τύπου σε εκφράσεις ανάθεσης, τελεστές και συνθήκες.

```
bears[4] = 42;
```

```
bears[8] = bears[2] + 42;
```

```
bears[ bears[4] ] = 2
```

Αρχικοποίηση Πίνακα

Παρεμφερής με την σύνταξη για αρχικοποίηση μεταβλητής:

```
int bears[100] = {  
    11, 25, 26, 31, 14, 13, 19, 3, 2, 19, 30, 7, 28, 9, 20, 19,  
    19, 1, 23, 15, 21, 18, 0, 25, 26, 20, 30, 29, 15, 29, 24, 9,  
    5, 20, 27, 13, 26, 14, 10, 27, 10, 3, 18, 31, 11, 19, 15, 9,  
    20, 15, 13, 31, 15, 9, 22, 22, 17, 30, 25, 14, 18, 0, 22, 13,  
    17, 2, 26, 10, 0, 9, 11, 10, 24, 2, 25, 18, 26, 31, 1, 18, 31,  
    1, 31, 9, 20, 15, 28, 17, 20, 14, 28, 11, 20, 14, 27, 11, 13,  
    6, 26, 31  
}
```

Εύρεση Μέγιστου Στοιχείου σε Πίνακα

Θέλουμε μια συνάρτηση `find_max` που να παίρνει έναν πίνακα 100 στοιχείων και να γυρίζει το μέγιστο:

Εύρεση Μέγιστου Στοιχείου σε Πίνακα

Θέλουμε μια συνάρτηση `find_max` που να παίρνει έναν πίνακα 100 στοιχείων και να γυρίζει το μέγιστο:

```
int find_max(int bears[100]) {  
    int i, max = bears[0];  
    for(i = 1; i < 100; i++) {  
        if (bears[i] > max) max = bears[i];  
    }  
    return max;  
}
```

Εντοπισμός Θέσεων Μνήμης Στοιχείου Πίνακα

Σε ποια διεύθυνση μνήμης βρίσκεται το στοιχείο (τύπου int) `bears[2]`;

$$\text{Αρχή του πίνακα} + 2 * \text{sizeof(int)} = 4 + 2 * 4 = 12$$

Μνήμη

Bytes 0-3

b0	b1	b2	b3
----	----	----	----

Bytes 4-7

b4	b5	b6	b7
----	----	----	----

Bytes 8-11

b8	b9	b10	b11
----	----	-----	-----

Bytes 12-15

b12	b13	b14	b15
-----	-----	-----	-----

...

Bytes 400-403

b400	b401	b402	b403
------	------	------	------

Το στοιχείο
`bears[2]` ξεκινάει
από το 12ο byte
της μνήμης

Δήλωση Πινάκων Διαφορετικών Τύπων

Πίνακες μπορούν να οριστούν για όλους τους τύπους της C. Παράδειγμα:

```
int a[1024];
```

```
char b[2048];
```

```
double c[512];
```

Ποιος από τους παραπάνω πίνακες καταλαμβάνει περισσότερη μνήμη;

Πίνακας Χαρακτήρων (String)

Ένας πίνακας από χαρακτήρες λέγεται και **αλφαριθμητικό / συμβολοσειρά (string)**. Λόγω της συχνής χρήσης τους, έχουμε αρκετές συντομεύσεις για αυτούς (θα δούμε και σε επόμενα μαθήματα). Οι τρεις παρακάτω δηλώσεις είναι ισοδύναμες:

```
char hello[] = {'H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd', '\n', '\0'};
```

```
char hello[] = {72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100, 10, 0};
```

```
char hello[] = "Hello World\n";
```

Προσοχή: τα string
τερματίζονται πάντα με
το null byte

'H'	'e'	'l'	'l'	'o'	' '	'W'	'o'	'r'	'l'	'd'	'\n'	'\0'
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------

Μπορώ να αναθέσω τα στοιχεία ενός πίνακα σε άλλον;

```
int a[3] = {1, 2, 3};
```

```
int b[3] = {4, 5, 6};
```

```
b = a;
```



Δεν επιτρέπεται στην C!

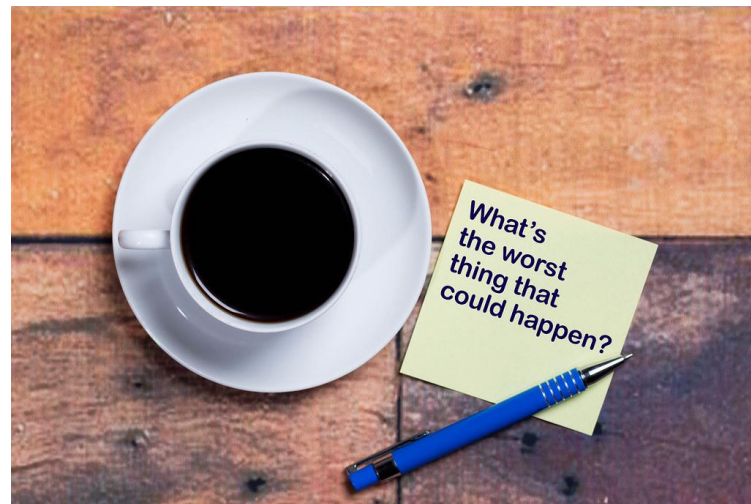
Που είναι χρήσιμοι οι πίνακες;

Είναι η βασικότερη **δομή δεδομένων (data structure)** στην πλειοψηφία των γλωσσών προγραμματισμού

- Μοντελοποίηση συνόλου τιμών ίδιου τύπου
- Μαζική δέσμευση και χρήση μνήμης με μια δήλωση
- Άμεση αποθήκευση, προσπέλαση και μετατροπή δεδομένων

Τι θα συμβεί αν προσπελάσω εκτός ορίων πίνακα;

- Υπερχείλιση (overflow) - π.χ. `bears[100]`
- Υποχείλιση (underflow) - π.χ. `bears[-1]`
- Το standard της γλώσσας κατηγοριοποιεί αυτήν την χρήση ως "undefined behavior"
- Στην πράξη αυτό σημαίνει ότι το πρόγραμμά μας θα κρασάρει (Segmentation Fault) ή ακόμα χειρότερα θα μας χακάρουν



Για την επόμενη φορά

- Σε αυτήν και την επόμενη διάλεξη θα καλύψουμε έννοιες από τις σελίδες 73-103 από τις σημειώσεις του κ. Σταματόπουλου.
- [Array \(data type\)](#) and as a [datastructure](#)
- [Array Programming](#)
- Play with [C Strings](#)
- [Buffers](#) & [Memoization](#)

Ευχαριστώ και καλή Σαββατοκύριακο εύχομαι!
Keep Coding ;)