

Διάλεξη 8 - Πίνακες

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Εισαγωγή στον Προγραμματισμό

Θανάσης Αυγερινός

Ανακοινώσεις / Διευκρινήσεις

- Βγήκε η εργασία #1, ημερομηνία παράδοσης: 6 Δεκεμβρίου, 23:59
 - a. Δεν υποβάλλουμε εργασία χωρίς σχόλια!
 - b. Τα χρονικά όρια ισχύουν για όλες τις εισόδους

Την προηγούμενη φορά

- Παραδείγματα Επαναληπτικών Δομών
- Άλλες Δομές Ελέγχου
- Επίλυση Προβλημάτων

Σήμερα

- Πίνακες (Arrays)

The image shows handwritten mathematical notes on a chalkboard. At the top left, there is a plot of a square wave function $f(t)$ with period $L=2$. The function is 1 for $t \in [0, 1]$ and -1 for $t \in [1, 2]$. The horizontal axis is labeled t and has tick marks at -1, 1, 2, 3, 4. Ellipses indicate the periodic nature of the function.

To the right of the square wave is a plot of the magnitude spectrum $\sqrt{a(\omega)^2 + b(\omega)^2}$ versus ω . The spectrum shows discrete lines at $\omega = 0, \pm\pi, \pm 2\pi, \dots$, with the amplitude decreasing as $|\omega|$ increases.

The Fourier series formula is written as:

$$f(t) = a_0 + \sum_{n=1}^{\infty} \left[a_n \cdot \cos\left(\frac{n\pi t}{L}\right) + b_n \cdot \sin\left(\frac{n\pi t}{L}\right) \right]$$
$$= a_0 + a_1 \cdot \cos\left(\frac{\pi t}{L}\right) + b_1 \cdot \sin\left(\frac{\pi t}{L}\right) + a_2 \cdot \cos\left(\frac{2\pi t}{L}\right) + b_2 \dots$$

Below the formula, there is a plot of a cosine wave $y = \cos(x)$ with period $L=2$. The horizontal axis is labeled t and has a tick mark at π .

Next to the cosine wave plot, the following calculations are shown:

$$L=2 \Rightarrow L=1 \quad a_n \approx \frac{1}{n^2}$$
$$b_n = \emptyset$$

The calculation for the average value a_0 is shown as:

$$a_0 = \frac{1}{2L} \int_{-L}^L f(t) dt = \frac{1}{2} \int_{-1}^1 f(t) dt$$
$$= \frac{1}{2} \int_{-1}^0 f(t) dt + \frac{1}{2} \int_0^1 f(t) dt$$
$$= \frac{1}{2} \int_{-1}^0 -1 dt + \frac{1}{2} \int_0^1 1 dt$$
$$= \frac{1}{2} [-t]_{-1}^0 + \frac{1}{2} [t]_0^1$$
$$= -\frac{1}{2} + \frac{1}{2} = \emptyset$$

On the right side of the chalkboard, there are two mappings:

$$a_n \rightarrow a(\omega)$$
$$b_n \rightarrow b(\omega)$$

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	1 DAY	2 HOURS	30 MINUTES	4 MINUTES	1 MINUTE	5 SECONDS
	5 SECONDS	5 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]`

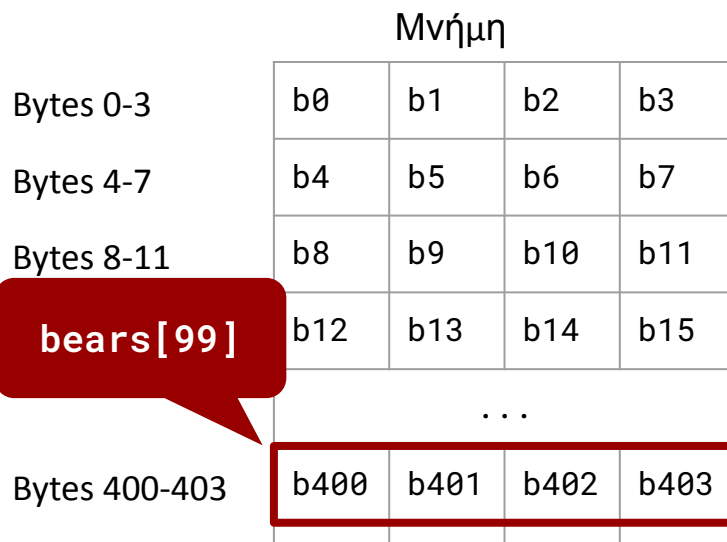
	Μνήμη			
bears[1]	b0	b1	b2	b3
Bytes 4-7	b4	b5	b6	b7
Bytes 8-11	b8	b9	b10	b11
Bytes 12-15	b12	b13	b14	b15
	...			
Bytes 400-403	b400	b401	b402	b403

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 ενός πίνακα 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			
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

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, 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!

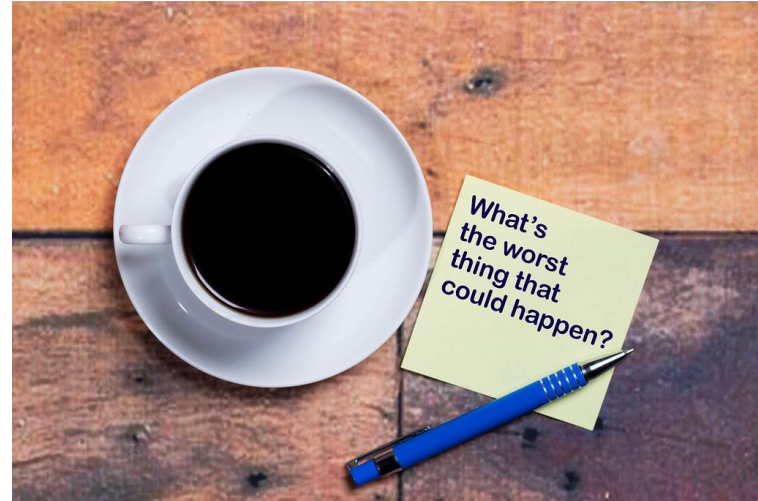
Που είναι χρήσιμοι οι πίνακες;

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



Θέλω μια συνάρτηση που να δέχεται έναν πίνακα 100 ακεραίων και να επιστρέφει τον μέσο όρο. Πως;

Θέλω μια συνάρτηση που να δέχεται έναν πίνακα 100 ακεραίων και να επιστρέφει τον μέσο όρο. Πως;

```
int average(int grades[100]) {  
    int i, sum = 0;  
    for(i = 0; i < 100; i++) {  
        sum += grades[i];  
    }  
    return sum / 100;  
}
```

Θέλω μια συνάρτηση που να δέχεται έναν πίνακα 100 ακεραίων και έναν ακέραιο και να γυρνάει την θέση του στοιχείου αν το βρήκε ή -1. Πως;

Θέλω μια συνάρτηση που να δέχεται έναν πίνακα 100 ακεραίων και έναν ακέραιο και να γυρνάει την θέση του στοιχείου αν το βρήκε ή -1. Πως;

```
int find(int haystack[100], int needle) {  
    int i;  
    for(i = 0; i < 100; i++) {  
        if (haystack[i] == needle) {  
            return i;  
        }  
    }  
    return -1;  
}
```

Θέλω μια συνάρτηση atoi που να παίρνει ένα πίνακα χαρακτήρων (μόνο ψηφία) και να επιστρέφει έναν ακέραιο. Πως;

Θέλω μια συνάρτηση `atoi` που να παίρνει ένα πίνακα χαρακτήρων (μόνο ψηφία) και να επιστρέφει έναν ακέραιο. Πως;

```
int atoi(char digits[]) {  
    int result = 0;  
    for(int i = 0; digits[i]; i++) {  
        result = 10 * result + digits[i] - '0';  
    }  
    return result;  
}
```

Τι μπορεί να πάει στραβά με αυτήν την συνάρτηση;

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 σαβκο εύχομαι!
Keep Coding ;)