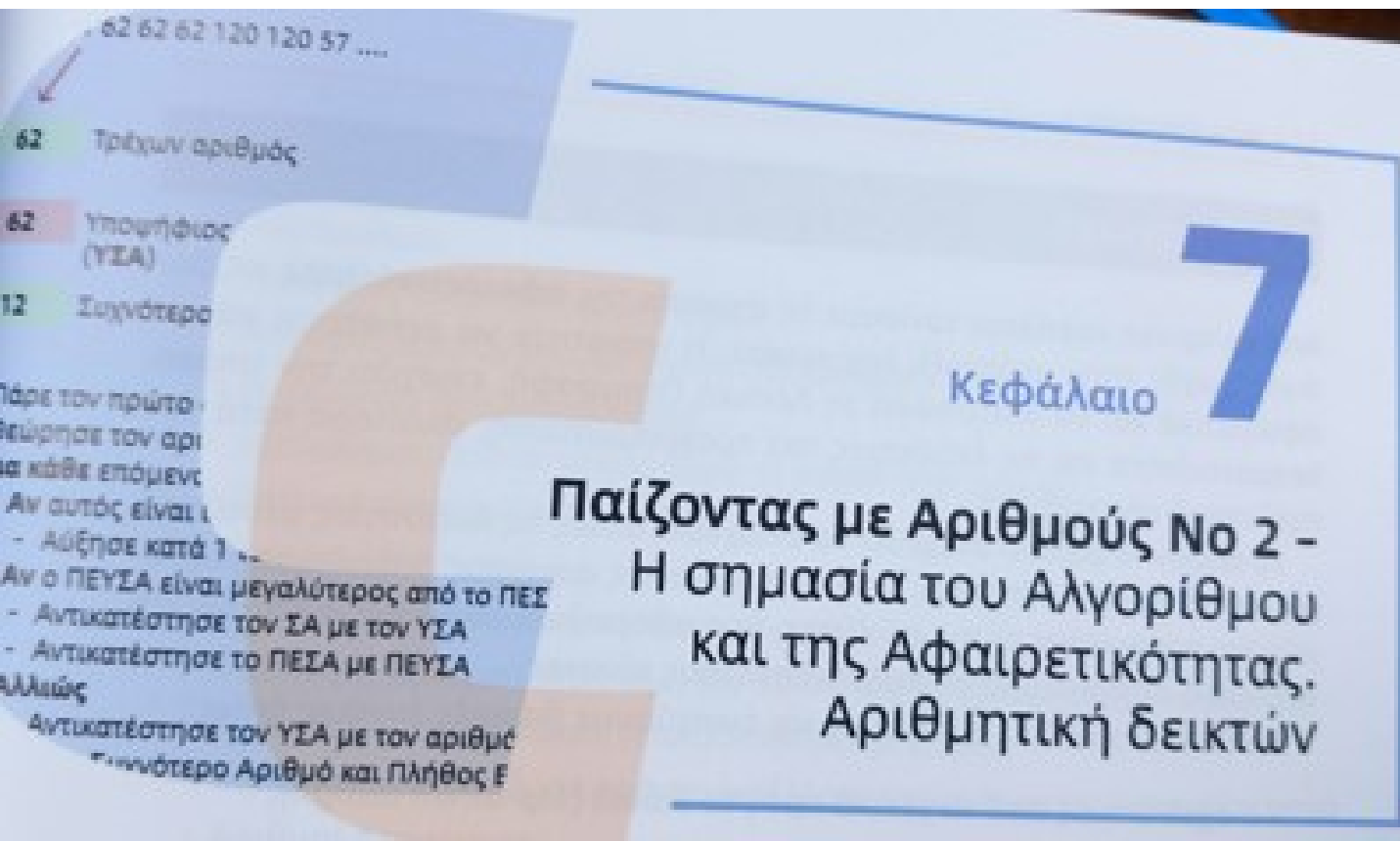


Εισαγωγή στην Τέχνη του Προγραμματισμού (με τη C) Παίζοντας με Αριθμούς

Η σημασία του Αλγορίθμου και της Αφαιρετικότητας. Αριθμητική δεικτών.

<https://sites.google.com/view/i2art-of-programming>



Πηγή: [Μια Εισαγωγή στην Τέχνη του Προγραμματισμού](#)

Algorithm
Abstraction
Pointer Arithmetic

Στόχος Ενότητας

- Στόχος του κεφαλαίου 7 είναι:
 - να αξιοποιήσει την γνώση και τις πρακτικές που αποκτήθηκαν σε προηγούμενες Δραστηριότητες,
 - να εξοικειώσει τον αναγνώστη με την **διαδικασία ανάπτυξης πιο σύνθετων και πιο απαιτητικών εφαρμογών** στις οποίες **ο σχεδιασμός του αλγορίθμου** ανάγεται στο πλέον σημαντικό βήμα για την ανάπτυξη **αρθρωτού κώδικα**.
- Προς την κατεύθυνση αυτή:
 - τονίζει τη σημασία της σταδιακής ανάπτυξης του αλγορίθμου μέσα από μια επαναληπτική διαδικασία για την **αντιμετώπιση της πολυπλοκότητας**, και,
 - εισάγει τις τεχνικές της τμηματοποίησης (**partitioning**) και της από το γενικό προς το ειδικό ανάπτυξης (**top-down**).

Η ικανότητα να σκέφτεται κανείς αφαιρετικά και να διατυπώνει τη Λεκτική Περιγραφή, ενισχύει την αποτελεσματικότητα και τις δεξιότητες του προγραμματιστή, ιδιαίτερα κατά τον σχεδιασμό αλγορίθμων.

- **εισάγει** και ενισχύει βασικές έννοιες όπως:
 - ο σχεδιασμός από το γενικό προς το ειδικό (top-down design),
 - ο αρθρωτός προγραμματισμός με χρήση συναρτήσεων,
 - η αριθμητική δεικτών, και
 - οι τεχνικές σχεδίασης αλγορίθμων.
- **εμβαθύνει** σε πιο προχωρημένα θέματα,
 - όπως η κλήση με αναφορά και η αναδρομικότητα, μέσα από το ενδιαφέρον παράδειγμα της εύρεσης αριθμών Armstrong.
- **προσφέρει** πρακτική εμπειρία στον σχεδιασμό και την υλοποίηση αλγορίθμων,
- **ενθαρρύνει**
 - την επαναληπτική ανάπτυξη για τη διαχείριση της πολυπλοκότητας,
 - τη σταδιακή διατύπωση του αλγορίθμου με τη μορφή Λεκτικής Περιγραφής, και
 - τη βελτιστοποίηση του χρόνου εκτέλεσης του προγράμματος.

Hands-on (Δράση 7-5 Αφαιρετική αναπαράσταση)

Αναπτύξτε ένα πρόγραμμα σύμφωνα με το οποίο το σύστημα θα διαβάζει μία ακολουθία από n αριθμούς και θα εμφανίζει:

- α) τον αριθμό που εμφανίστηκε στην ακολουθία περισσότερες συνεχόμενες φορές, και
- β) το πλήθος των συνεχόμενων εμφανίσεων του.

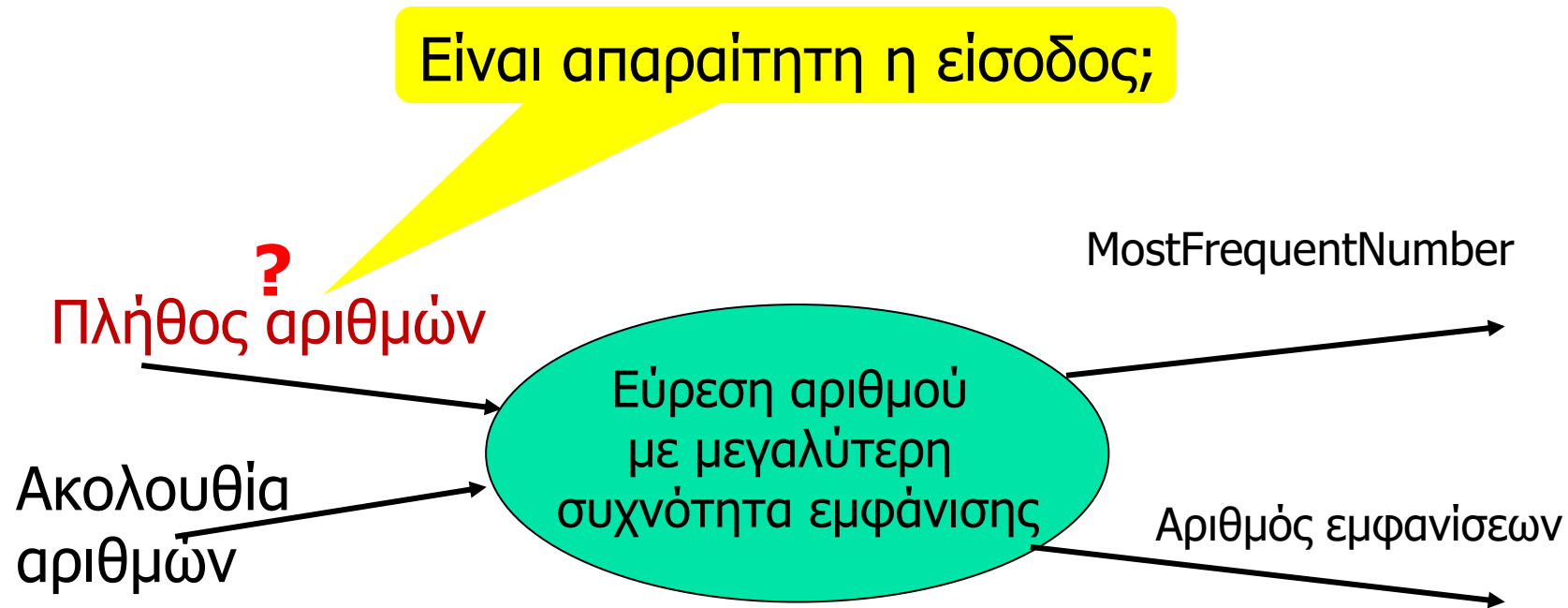


Δώστε την αφαιρετική αναπαράσταση και ορίστε μεταβλητές εισόδου και εξόδου χρησιμοποιώντας ονόματα που ορίζουν την πληροφορία που αναπαριστούν (αναγνωσιμότητα κώδικα)

Hands-on Αφαιρετική αναπαράσταση

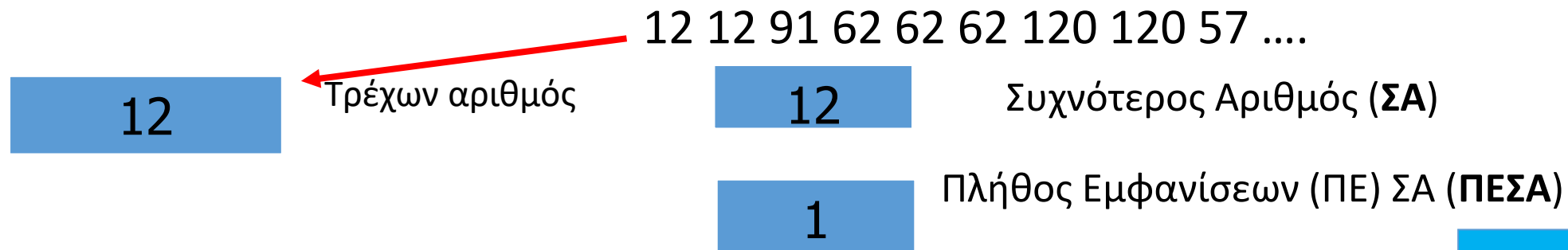
Αναπτύξτε ένα πρόγραμμα σύμφωνα με το οποίο το σύστημα θα διαβάζει μία ακολουθία από n αριθμούς και θα εμφανίζει:

- α) τον αριθμό που εμφανίστηκε στην ακολουθία περισσότερες συνεχόμενες φορές, και
- β) το πλήθος των συνεχόμενων εμφανίσεων του.



Hands-on Λεκτική Περιγραφή – 1^η Εκδοχή

Αναπτύξτε ένα πρόγραμμα σύμφωνα με το οποίο το σύστημα θα διαβάσει μία ακολουθία από n αριθμούς και θα εμφανίζει: α) τον αριθμό που εμφανίστηκε στην ακολουθία περισσότερες συνεχόμενες φορές, και β) το πλήθος των συνεχόμενων εμφανίσεων του.



Αναγνωσιμότητα
(Readability)
Απλότητα
(Simplicity)

Πάρε τον πρώτο αριθμό

Θεώρησε τον αριθμό ΣΑ με ΠΕ 1

Για κάθε επόμενο αριθμό της ακολουθίας

- **Αν** αυτός είναι ίσος με τον ΣΑ
 - Αύξησε κατά 1 το ΠΕΣΑ
- **Αλλιώς**
 - Αντικατέστησε τον ΣΑ με τον τρέχοντα αριθμό

Εμφάνισε Συχνότερο Αριθμό και Πλήθος Εμφανίσεων

Συνεχίστε τον έλεγχο για διάφορα
σενάρια ακολουθίας

Εναλλακτικά αναπτύξτε τον
κώδικα και ελέγξτε διάφορα
σενάρια ακολουθίας

Hands-on Λεκτική Περιγραφή – 2^η Έκδοχή

Αναπτύξτε ένα πρόγραμμα σύμφωνα με το οποίο το σύστημα θα διαβάζει μία ακολουθία από η αριθμούς και θα εμφανίζει: α) τον αριθμό που εμφανίστηκε στην ακολουθία περισσότερες συνεχόμενες φορές, και β) το πλήθος των συνεχόμενων εμφανίσεων του.

62	Τρέχον αριθμός		12 12 91 62 62 62 120 120 57
62	Υποψήφιος Συχνότερος Αριθμός (ΥΣΑ)	3	Πλήθος Εμφανίσεων ΥΣΑ (ΠΕΥΣΑ)
12	Συχνότερος Αριθμός (ΣΑ)	2	Πλήθος Εμφανίσεων ΣΑ (ΠΕΣΑ)
....			

Για κάθε επόμενο αριθμό της ακολουθίας

- **Αν** αυτός είναι ίσος με τον ΥΣΑ
 - Αύξησε κατά 1 το ΠΕΥΣΑ
 - **Αν** ο ΠΕΥΣΑ είναι μεγαλύτερος από το ΠΕΣΑ
 - Αντικατέστησε τον ΣΑ με τον ΥΣΑ
 - Αντικατέστησε το ΠΕΣΑ με ΠΕΥΣΑ
- **Αλλιώς**
 - Αντικατέστησε τον ΥΣΑ με τον αριθμό

Εμφάνισε ΣΑ και ΠΕΣΑ

Βρείτε λάθη
Βελτιώστε τον

Hands-on Πηγαίος Κώδικας



Για κάθε επόμενο αριθμό της ακολουθίας

- **Αν** αυτός είναι ίσος με τον ΥΣΑ
 - Αύξησε κατά 1 το ΠΕΥΣΑ
 - **Αν** ο ΠΕΥΣΑ είναι μεγαλύτερος από το ΠΕΣΑ
 - Αντικατέστησε τον ΣΑ με τον ΥΣΑ
 - Αντικατέστησε το ΠΕΣΑ με ΠΕΥΣΑ

■ Αλλιώς

- Αντικατέστησε τον ΥΣΑ με τον αριθμό

Εμφάνισε ΣΑ και ΠΕΣΑ

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "..\..\i2p.h"
4
5  int num, numOfTerms;
6  int ysa, peysa=0;
7  int sa,pesa=0;
8
9  int main(int argc, char *argv[]) {
10     int i;
11     printf("MostFrequentNumber\n");
12     numOfTerms = getInt("Dose plithos q");
13     num=getInt("Dose arithmo:");
14     ysa = num;
15     peysa=1;
16     for(i=0;i<numOfTerms;i++){
```

1^η έκδοση του πηγαίου κώδικα μόνο με main

Προχωρήστε σε μια 2^η modular έκδοση

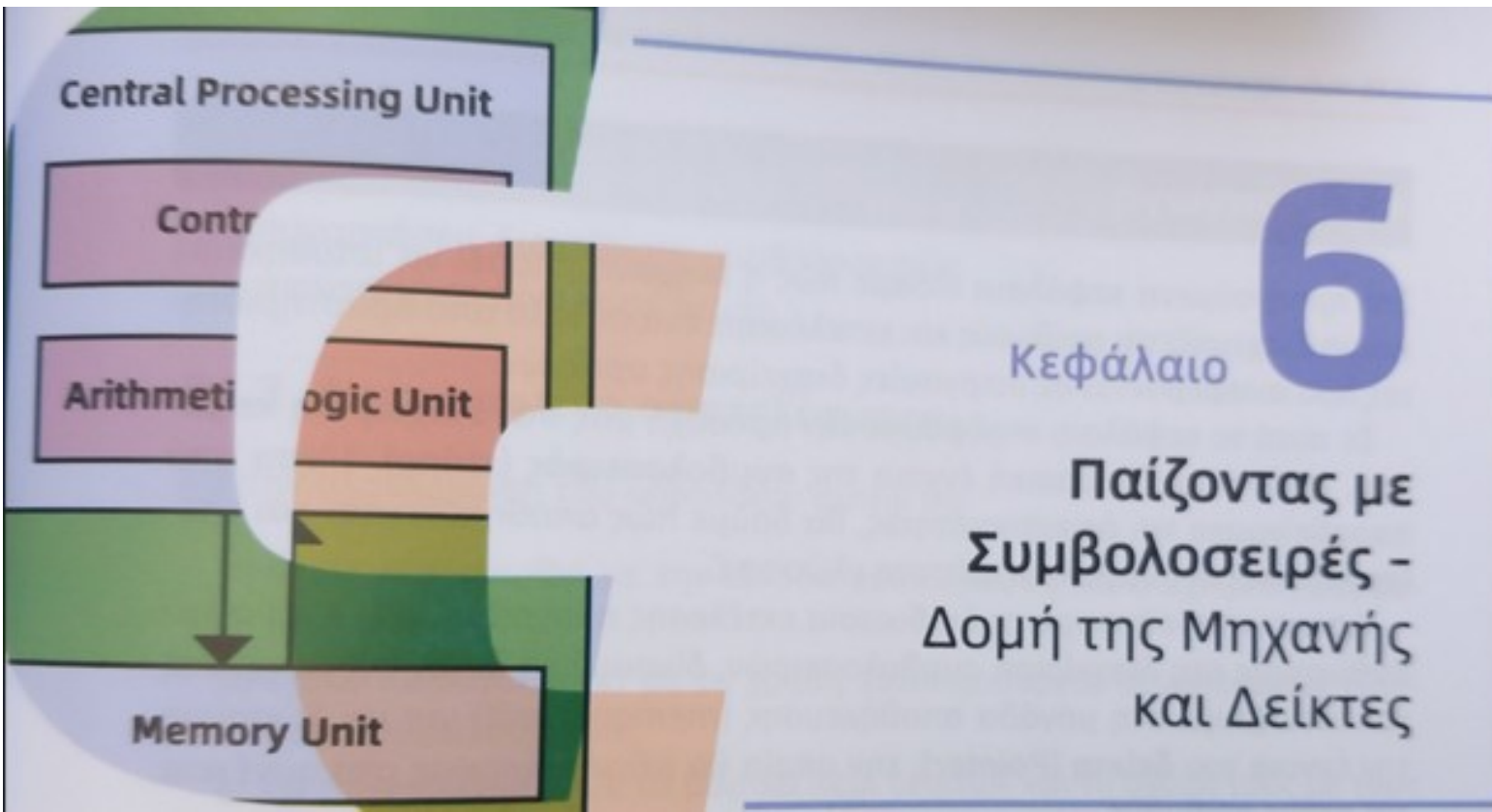
Εισαγωγή στον Προγραμματισμό (Introduction to Programming)

(CEID_NY131)

Παίζοντας με Αλφαριθμητικά

Δομή της Μηχανής και Δείκτες

<https://sites.google.com/view/i2art-of-programming>



Πηγή: [Μια Εισαγωγή στην Τέχνη του Προγραμματισμού](#)

Strings - Pointers

Στόχος Ενότητας

- **Εισάγει** τη βασική έννοια της συμβολοσειράς (string) που μας επιτρέπει να χρησιμοποιούμε τη Μηχανή για διαχείριση (αποθήκευση και επεξεργασία) κειμένου.
- Να δώσει τη δομή της Μηχανής για να κατανοήσουμε τη διαδικασία εκτέλεσης προγράμματος.
- Εισάγει την έννοια του Δείκτη (Pointer) για πρόσβαση στην μονάδα αποθήκευσης (memory unit) για πιο ευέλικτη διαχείριση συμβολοσειρών.

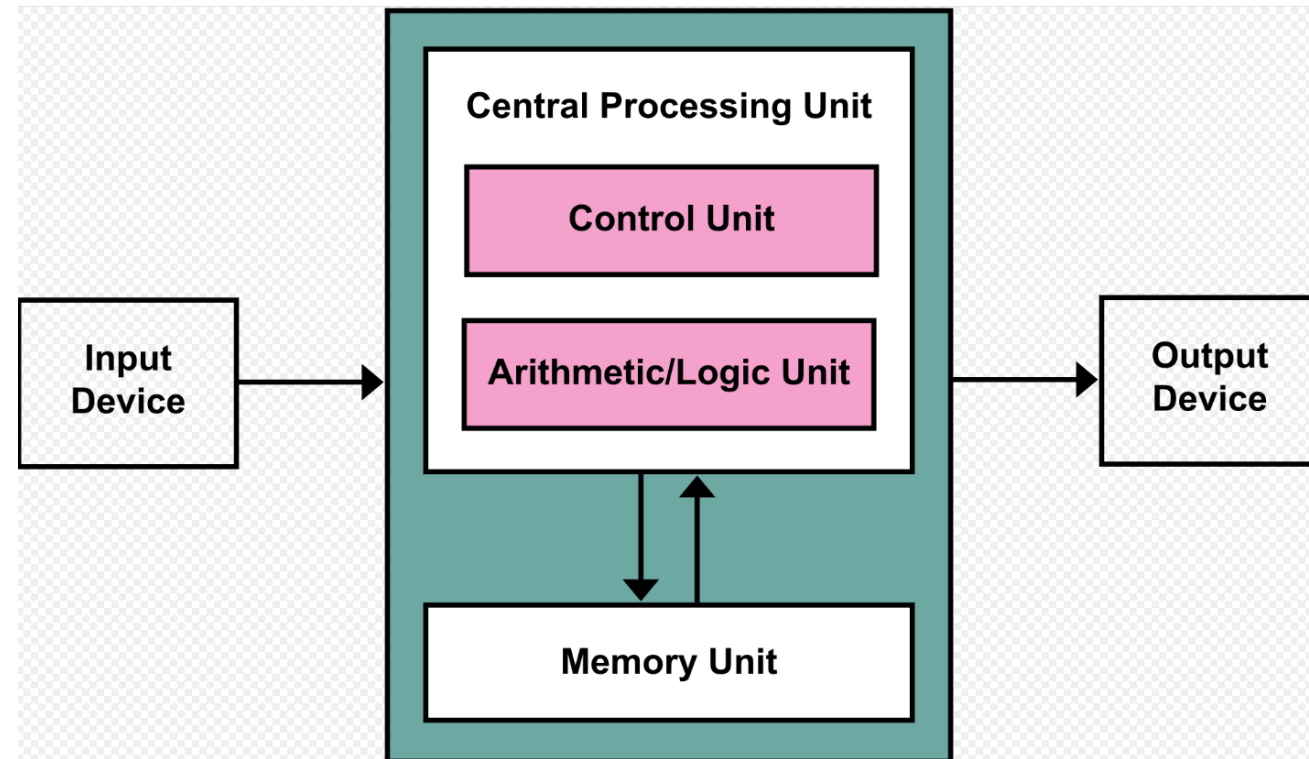
Προχωράμε σε πιο βαθιά νερά για να δούμε τη σχέση της σημειογραφίας του Δείκτη με τη σημειογραφία του Πίνακα. Πίνακες και Δείκτες είναι από τα πιο ισχυρά στοιχεία της C.

- **Η δομή της «Μηχανής»**
 - Von Neumann Architecture, Γλώσσα Μηχανής
- **Δραστηριότητα 6.1 - AsciiCode**
 - Ο τύπος χαρακτήρα (char) και ο ASCII κώδικας
 - Μονάδες αποθήκευσης, Διευθύνσεις και Δείκτες
- **Μονάδες αποθήκευσης, Διευθύνσεις και Δείκτες**
- **Δραστηριότητα 6.2 - StringHandling**
 - Συμβολοσειρά (String) και αλφαριθμητικό
 - Αντιστοίχιση σημειογραφίας πίνακα με σημειογραφία Δείκτη
 - Συναρτήσεις διαχείρισης συμβολοσειρών
- **Δραστηριότητα 6.3 - GetStringVersions**
 - Ο Alex, η getStringV2 και το chatbot

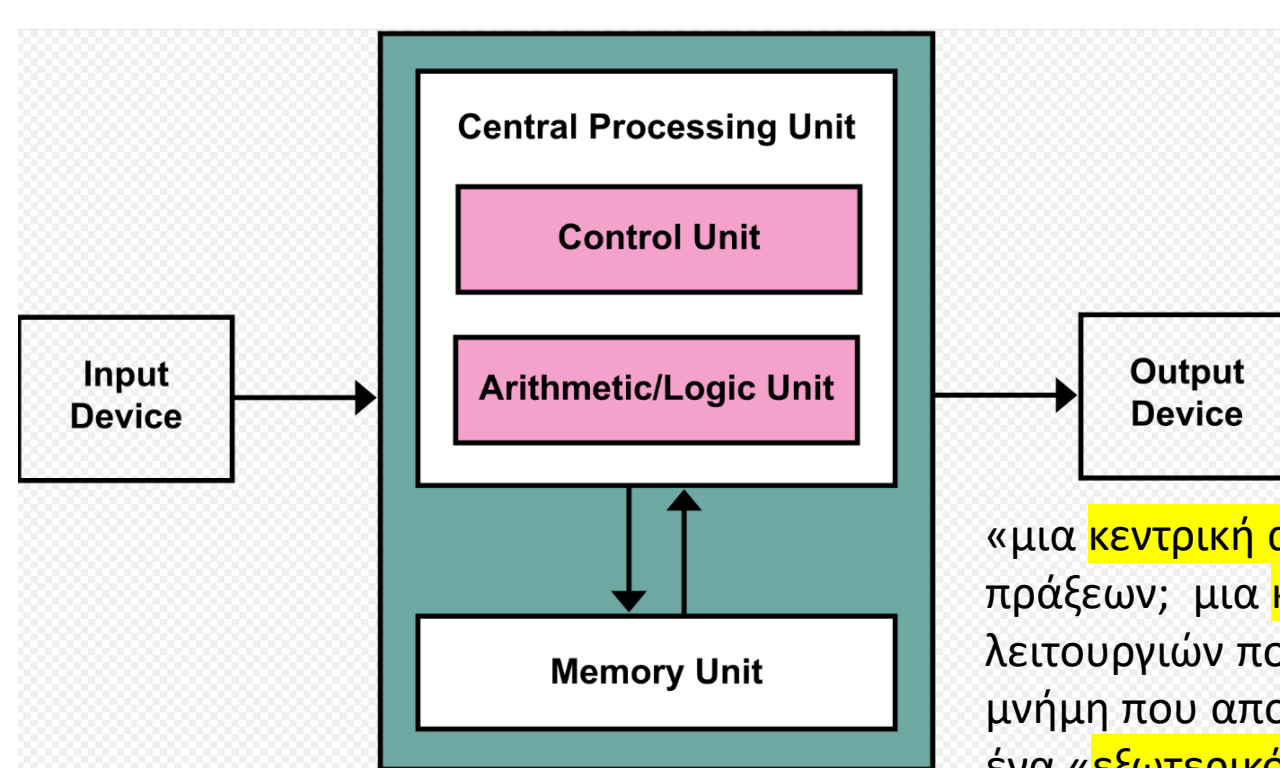
Δομή της «Μηχανής» - Von Neumann Architecture

“Η αρχιτεκτονική von Neumann είναι ένας σχεδιασμός υπολογιστή που χρησιμοποιεί μια ενιαία μονάδα μνήμης για την αποθήκευση εντολών και δεδομένων προγράμματος, μια έννοια γνωστή ως έννοια αποθηκευμένου προγράμματος. Σχεδιάστηκε από τον John von Neumann το 1945 και αποτελείται από μια μονάδα επεξεργασίας (συμπεριλαμβανομένης μιας Αριθμητικής Λογικής Μονάδας (ALU) και μιας Μονάδας Ελέγχου), μνήμης και συσκευών εισόδου/εξόδου....”

wikipedia



Δομή της «Μηχανής» - Von Neumann Architecture



- κεντρική μονάδα επεξεργασίας (central processing unit)
- μονάδα μνήμης (memory unit)
- μονάδα εισόδου (Input Unit)
- μονάδα εξόδου (Output Unit)

«μια **κεντρική αριθμητική μονάδα** για την εκτέλεση αριθμητικών πράξεων; μια **κεντρική μονάδα ελέγχου** για την αλληλουχία των λειτουργιών που εκτελούνται από το μηχάνημα· μνήμη που αποθηκεύει δεδομένα και οδηγίες.

ένα «**εξωτερικό μέσο εγγραφής**» για την αποθήκευση εισόδου και εξόδου από το μηχάνημα; μηχανισμούς εισόδου και εξόδου για τη μεταφορά δεδομένων μεταξύ της μνήμης και του εξωτερικού μέσου εγγραφής.

... όλα συνδέονται με κοινόχρηστο δίαυλο . Αυτός ο σχεδιασμός επιτρέπει υπολογιστές γενικής χρήσης επιτρέποντας τη διαδοχική ανάκτηση, αποκωδικοποίηση και εκτέλεση εντολών, αν και δημιουργεί ένα σημείο συμφόρησης με την κοινή χρήση του διαύλου για δεδομένα και οδηγίες» wikipedia

Η Γλώσσα Μηχανής – Assembly – C

Η κεντρική μονάδα επεξεργασίας μπορεί να εκτελεί μόνο εντολές σε γλώσσα Μηχανής (machine language).

32-bit **x86** machine code

```
8B542408 83FA0077 06B80000
0000C383 FA027706 B8010000
00C353BB 01000000 B9010000
008D0419 83FA0376 078BD98B
C84AEBF1 5BC3
```

Assembly: η πρώτη **συμβολική γλώσσα** που δημιουργήθηκε για να διευκολυνθεί η συγγραφή προγραμμάτων.
Ήταν για πολλά χρόνια, και μέχρι να δημιουργηθεί η C, η γλώσσα για τη συγγραφή των λειτουργικών συστημάτων (Operating Systems – OS).

x86 assembly language using MASM syntax:

fib:

```
mov edx, [esp+8]
cmp edx, 0
ja @f
mov eax, 0
ret
```

```
@@:
cmp edx, 2
ja @f
mov eax, 1
ret
```

```
@@:
push ebx
mov ebx, 1
mov ecx, 1
```

@@:

```
lea eax, [ebx+ecx]
cmp edx, 3
jbe @f
mov ebx, ecx
mov ecx, eax
dec edx
```

jmp @b

@@:

```
pop ebx
ret
```


Οργάνωση Διάλεξης

- Η δομή της «Μηχανής»
 - Von Neumann Architecture, Γλώσσα Μηχανής
- **Δραστηριότητα 6.1 - AsciiCode**
 - Ο τύπος χαρακτήρα (char) και ο ASCII κώδικας
 - Μονάδες αποθήκευσης, Διευθύνσεις και Δείκτες
- **Μονάδες αποθήκευσης, Διευθύνσεις και Δείκτες**
- **Δραστηριότητα 6.2 - StringHandling**
 - Συμβολοσειρά (String) και αλφαριθμητικό
 - Αντιστοίχιση σημειογραφίας πίνακα με σημειογραφία Δείκτη
 - Συναρτήσεις διαχείρισης συμβολοσειρών
- **Δραστηριότητα 6.3 - GetStringVersions**
 - Ο Alex, η getStringV2 και το chatbot

Δραστηριότητα 6.1 - AsciiCode

Αναπτύξτε ένα πρόγραμμα με βάση το οποίο η Μηχανή θα τυπώνει τους χαρακτήρες του λατινικού αλφαβήτου (πεζά και κεφαλαία) και τα δέκα ψηφία, και για το καθένα από αυτά τον αντίστοιχο ascii κωδικό του.

➤ Η Δραστηριότητα έχει στόχο την εισαγωγή στις βασικές έννοιες που επιτρέπουν την αποθήκευση και διαχείριση χαρακτήρων από τη Μηχανή.

ASCII, stands for American Standard Code for Information Interchange. It is a **7-bit** character code where each individual bit represents a unique character. This page shows the **extended ASCII** table which is **based on the Windows-1252** character set which is an **8 bit** ASCII table with 256 characters and symbols. It includes all ASCII codes from standard ASCII, and it is a superset of **ISO 8859-1** in terms of printable characters. In the range 128 to 159 (hex 80 to 9F), ISO/IEC 8859-1 has invisible control characters, while Windows-1252 has writable characters. Windows-1252 is probably the most-used 8-bit character encoding in the world.

<https://www.ascii-code.com/>

Ο τύπος του χαρακτήρα και ο ASCII Κώδικας

Η C όπως κάθε γλώσσα προγραμματισμού έχει το δικό της **αλφάβητο**, το οποίο χρησιμοποιεί για το σχηματισμό λέξεων που αποτελούν το λεξιλόγιο της.

Καθώς η Μηχανή μπορεί να αποθηκεύσει στην μνήμη της μόνο αριθμούς δημιουργήθηκαν πίνακες αντιστοίχισης χαρακτήρων σε αριθμούς, όπως είναι ο πίνακας ASCII ...

κάθε
χαρακτήρας
έχει ένα
μοναδικό
αριθμητικό
κωδικό

55	067	37	00110111	7	7		Seven
56	070	38	00111000	8	8		Eight
57	071	39	00111001	9	9		Nine
58	072	3A	00111010	:	:	:	Colon
59	073	3B	00111011	;	;	;	Semicolon
60	074	3C	00111100	<	<	<	Less than (or op
61	075	3D	00111101	=	=	=	Equals
62	076	3E	00111110	>	>	>	Greater than (or
63	077	3F	00111111	?	?	?	Question mark
64	100	40	01000000	@	@	@	At sign
65	101	41	01000001	A	A		Uppercase A
66	102	42	01000010	B	B		Uppercase B
67	103	43	01000011	C	C		Uppercase C

<https://www.ascii-code.com/ASCII>

Μεταβλητή τύπου χαρακτήρα

```
char ch;
```

δηλώνει μεταβλητή με όνομα ch ως τύπου χαρακτήρα έχει ως αποτέλεσμα τη δέσμευση χώρου στην μνήμη για την αποθήκευση ενός χαρακτήρα

```
char ch1 = 'A' ;
```

δηλώνει μεταβλητή με όνομα ch1 και της αποδίδει ως αρχική τιμή τον χαρακτήρα A

```
void putchar(char ch) ;
```

Δήλωση της συνάρτησης putchar της τυπικής βιβλιοθήκης για την εμφάνιση στην κύρια έξοδο ενός χαρακτήρα

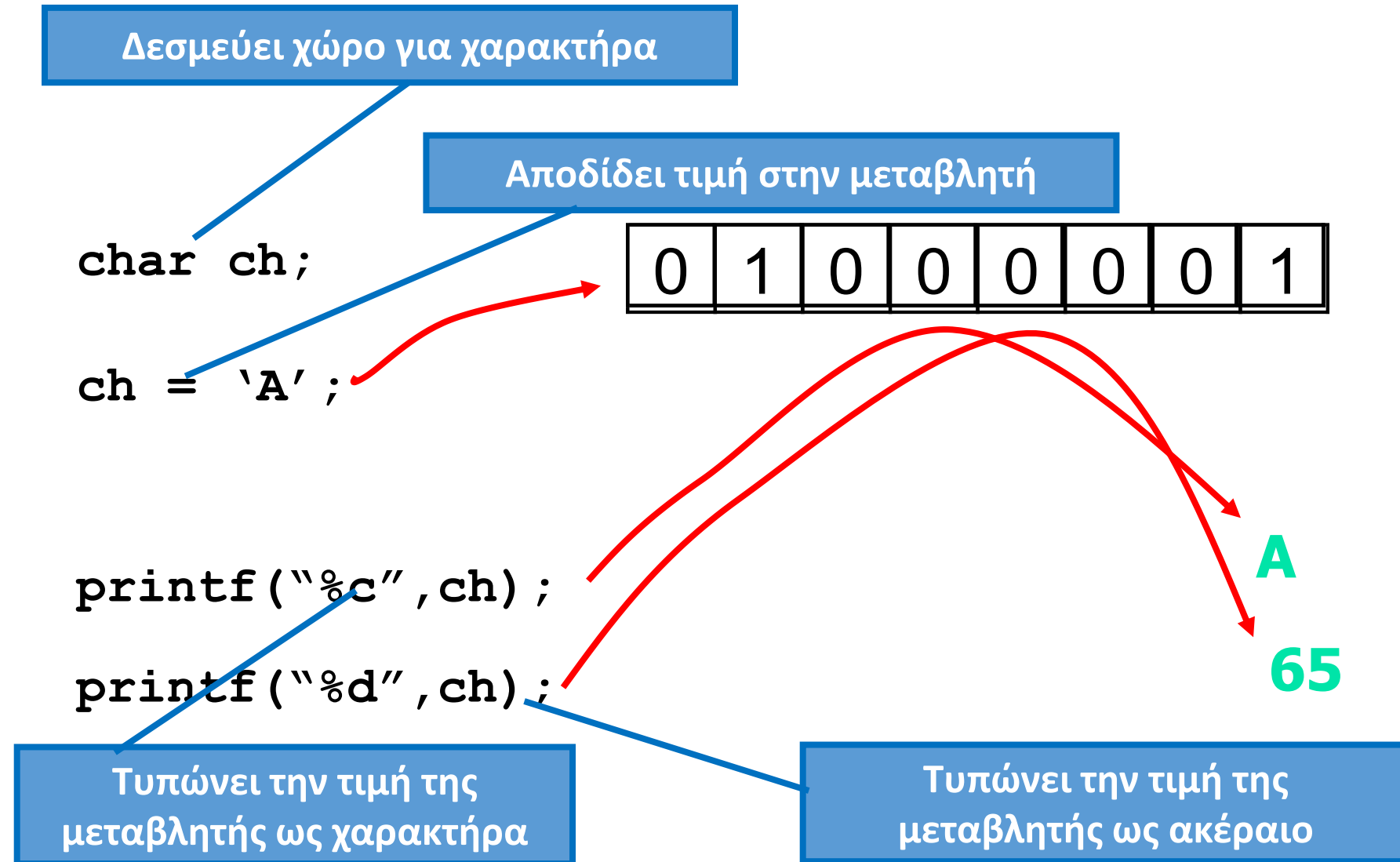
```
printf("ch1=%c \n", ch1) ;
```

Παράδειγμα χρήσης της printf για εμφάνιση στην κύρια έξοδο ενός χαρακτήρα

```
printf("ch1 as decimal =%d \n", ch1) ;
```

Παράδειγμα χρήσης της printf για εμφάνιση στην κύρια έξοδο του ascii κωδικού ενός χαρακτήρα

Ο τύπος του χαρακτήρα και ο ASCII Κώδικας



Πρόγραμμα: Εμφάνιση χαρακτήρα στη κύρια έξοδο

mainV1.c X

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include "../i2p.h"
4
5  char ch1='A';
6  char tab = '\t';
7  char newLine = '\n';
8
9  int main()
10 {
11     printf("CharHandlingV1!\n");
12     printf("ch1=%c. \n", ch1);
13     printf("ch1 as decimal =%d. \n", ch1);
14
15     printf("tab=%c. \n", tab);
16     printf("tab as decimal =%d. \n", tab);
17
18     printf("newLine=%c. \n", newLine);
19     printf("newLine as decimal =%d. \n", newLine);
20
21     return 0;
22 }
```

C:\Code\I2P\Chapter6\CharHi

CharHandlingV1!

ch1=A.

ch1 as decimal =65.

tab= .

tab as decimal =9.

newLine=

.

newLine as decimal =10.

Hands-on (AsciiCode)

Δράση 6.1 Εμφάνιση ASCII κώδικα (AsciiCodeV1)

Αναπτύξτε μια πρώτη έκδοση του προγράμματος AsciiCode με βάση το οποίο η Μηχανή θα τυπώνει τον ascii κώδικα. Από την έξοδο του προγράμματος θα δείτε τις θέσεις των πεζών και κεφαλαίων λατινικών γραμμάτων καθώς και αυτή των ψηφίων και με βάση αυτό θα διαμορφώσετε την 2η έκδοση.

Δράση 6.2 Εμφάνιση ASCII κώδικα (AsciiCodeV2)

Αναπτύξτε ένα πρόγραμμα με βάση το οποίο η Μηχανή θα τυπώνει τους χαρακτήρες του λατινικού αλφαβήτου (πεζά και κεφαλαία) και τα δέκα ψηφία, και για το καθένα από αυτά τον αντίστοιχο ascii κωδικό του.



Αναπτύξτε τις 2 εκδόσεις της
Δραστηριότητας **AsciiCode**

Ελέγξτε αν υλοποιούν σωστά την λειτουργικότητα

Είσοδος χαρακτήρα – getchar

```
char getchar(void) ;
```

Δήλωση της συνάρτησης getchar της τυπικής βιβλιοθήκης για είσοδο χαρακτήρα από την κύρια έξοδο

```
ch = getchar() ;
```

Παράδειγμα χρήσης της getchar για είσοδο από την κύρια είσοδο ενός χαρακτήρα και απόδοσης του ως τιμής σε μεταβλητή χαρακτήρα

Hands-on (AsciiCode)

Δράση 6.3 Διαχείριση χαρακτήρων (CharHandling)

Αναπτύξτε ένα πρόγραμμα με βάση το οποίο η Μηχανή θα διαβάζει από τη κύρια είσοδο, χαρακτήρα προς χαρακτήρα, μία λέξη 6 πεζών χαρακτήρων, θα αποθηκεύει αυτή καθώς και τη λέξη με τα αντίστοιχα κεφαλαία και στη συνέχεια θα τις τυπώνει.



**Αναπτύξτε το πρόγραμμα κάνοντας
χρήση των συναρτήσεων `getchar()` και
`putchar()`**

Οργάνωση Διάλεξης

- Η δομή της «Μηχανής»
 - Von Neumann Architecture, Γλώσσα Μηχανής
- Δραστηριότητα 6.1 - AsciiCode
 - Ο τύπος χαρακτήρα (char) και ο ASCII κώδικας
 - Μονάδες αποθήκευσης, Διευθύνσεις και Δείκτες
- Μονάδες αποθήκευσης, Διευθύνσεις και Δείκτες
- Δραστηριότητα 6.2 - StringHandling
 - Συμβολοσειρά (String) και αλφαριθμητικό
 - Αντιστοίχιση σημειογραφίας πίνακα με σημειογραφία Δείκτη
 - Συναρτήσεις διαχείρισης συμβολοσειρών
- Δραστηριότητα 6.3 - GetStringVersions
 - Ο Alex, η getStringV2 και το chatbot

Μονάδες μνήμης - Διεύθυνση

- **Bit**

0	1
---	---

 - αποτελεί το βασικό δομικό στοιχείο της μνήμης του υπολογιστή. Μπορεί να έχει μια εκ των δύο τιμών: 0 ή 1
- **Byte**

0	1	0	1	1	0	1	1
---	---	---	---	---	---	---	---

ANSI requirement
 - αποτελεί τη συνηθισμένη μονάδα αποθήκευσης. Αποτελείται από 8bits
- **Word**

0	1	0	1	1	0	1	1	1	1	0	1	1	0
---	---	---	---	---	---	---	---	---	---	---	---	---	---

 - αποτελεί την πληροφορία που μπορεί να διακινηθεί (I/O) σε ένα κύκλο μηχανής.
 - ταυτίζεται με το μέγεθος του data bus του επεξεργαστή
 - 8-bit, 16-bit, 32-bit, 64-bit microcomputers.

Κάθε Byte της κύριας μνήμης έχει μια **διεύθυνση**, που είναι ένας αύξων αριθμός από το πρώτο έως το τελευταίο byte της κύριας μνήμης.

Διεύθυνση, τελεστής άμεσης διεύθυνσης και Δείκτης

Η C μας δίνει τη δυνατότητα να πάρουμε, με τον **τελεστή άμεσης διεύθυνσης &**, τη διεύθυνση της μνήμης, που η δήλωση μιας μεταβλητής δέσμευσε.

Η έκφραση **&ch** έχει τιμή τη διεύθυνση του byte που δεσμεύθηκε στη κύρια μνήμη από τη δήλωση της μεταβλητής χαρακτήρα `ch`

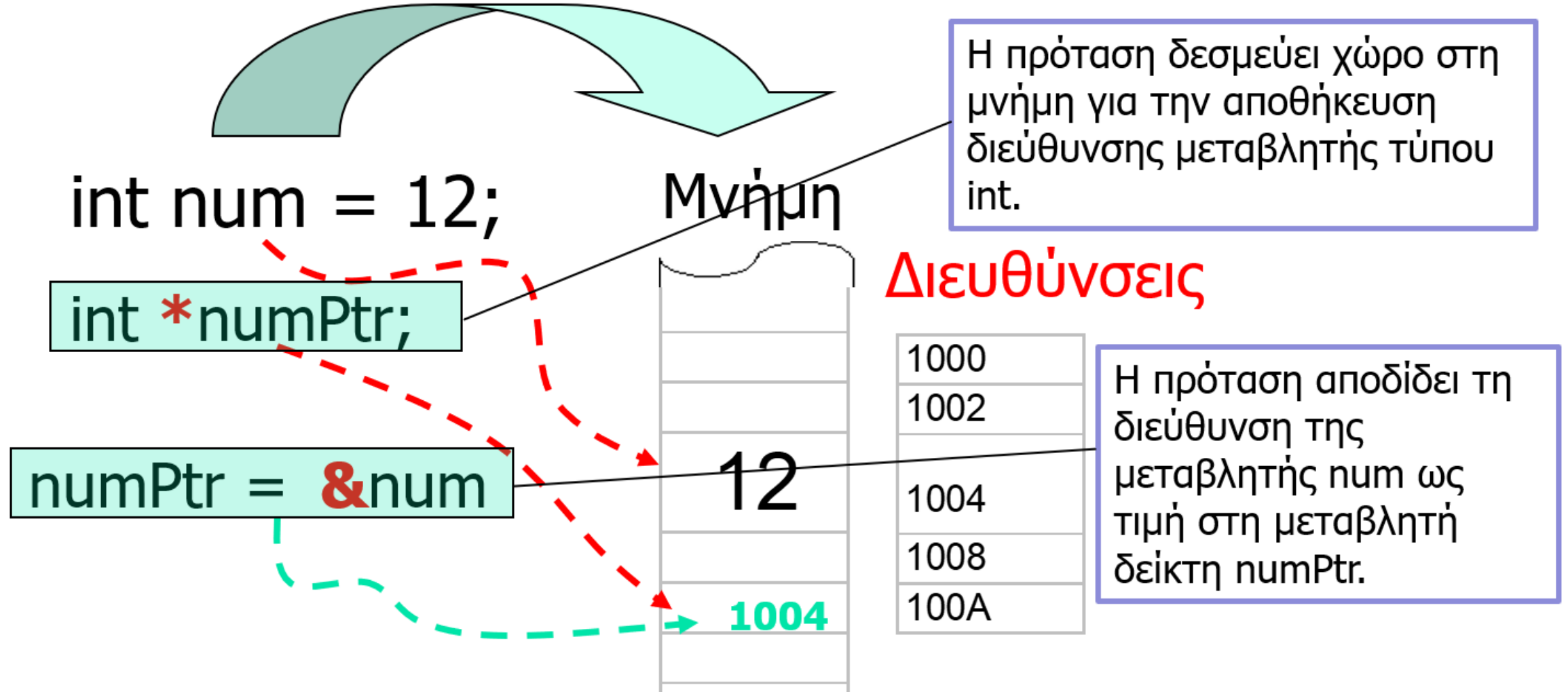
Τη διεύθυνση μιας μεταβλητής μπορούμε να την αποθηκεύσουμε σε μία άλλη μεταβλητή, η οποία θα πρέπει να δηλωθεί ως **μεταβλητή Δείκτη** (Pointer).

Για τη δήλωση μιας μεταβλητής Δείκτη χρησιμοποιούμε πριν από το όνομα της μεταβλητής τον τελεστή *****.

```
int *numPtr;
```

Δήλωση της μεταβλητής `numPtr` ως δείκτη (*) σε θέση μνήμης που δεσμεύτηκε για μεταβλητή τύπου ακεραίου (`int`).

Διεύθυνση, τελεστής άμεσης διεύθυνσης και Δείκτης



Hands-on (Pointers)

Αναπτύξτε ένα πρόγραμμα που θα επιδεικνύει :

- τη δήλωση μεταβλητής Δείκτη,
- την απόδοση τιμής σε αυτή ώστε να δείχνει σε ακέραια μεταβλητή που έχει δηλωθεί και στην οποία έχει αποδοθεί αρχική τιμή
- την εμφάνιση της τιμής της ακέραιας μεταβλητής κάνοντας χρήση της μεταβλητής δείκτη, και
- την εμφάνιση της τιμής που έχει η μεταβλητή δείκτη.



Αναπτύξτε το πρόγραμμα και ελέγξτε τη λειτουργικότητά του

Οργάνωση Διάλεξης

- Η δομή της «Μηχανής»
 - Von Neumann Architecture, Γλώσσα Μηχανής
- Δραστηριότητα 6.1 - AsciiCode
 - Ο τύπος χαρακτήρα (char) και ο ASCII κώδικας
 - Μονάδες αποθήκευσης, Διευθύνσεις και Δείκτες
- Μονάδες αποθήκευσης, Διευθύνσεις και Δείκτες
- Δραστηριότητα 6.2 - StringHandling
 - Συμβολοσειρά (String) και αλφαριθμητικό
 - Αντιστοίχιση σημειογραφίας πίνακα με σημειογραφία Δείκτη
 - Συναρτήσεις διαχείρισης συμβολοσειρών
- Δραστηριότητα 6.3 - GetStringVersions
 - Ο Alex, η getStringV2 και το chatbot

Δραστηριότητα 6.2 - StringHandling

Αναπτύξτε ένα πρόγραμμα με βάση το οποίο η Μηχανή θα διαβάζει από την είσοδο το όνομα του χρήστη και στη συνέχεια θα το τυπώνει με τους παρακάτω τρόπους:

α) κανονικά, β) με αντίστροφη σειρά, γ) ένα χαρακτήρα ανά γραμμή,

δ) όπως παρακάτω (για όνομα NIKOS)

NIKOS
NIKO
NIK
NI
N

ε) όπως παρακάτω

NIKOS
IKOS
KOS
OS
S

➤ Η Δραστηριότητα έχει στόχο την εισαγωγή, την αποθήκευση και τη διαχείριση συμβολοσειρών από τη Μηχανή.

Συμβολοσειρά (String) και αλφαριθμητικό

Συμβολοσειρά **είναι** μια ακολουθία συμβόλων που ανήκουν σε ένα πεπερασμένο σύνολο, το οποίο ονομάζεται αλφάβητο.

Αν η συμβολοσειρά αποτελείται μόνο από αλφαριθμητικούς χαρακτήρες και αριθμούς τότε ονομάζεται αλφαριθμητικό.

Η C δεν διαθέτει τύπο για συμβολοσειρά, αλλά χειρίζεται τις **συμβολοσειρές ως πίνακες χαρακτήρων**.

Το αλφάβητο της γλώσσας C σύμφωνα με το ANSI (American Standard Code for Information Interchange) πρότυπο αποτελείται από 96 χαρακτήρες.

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9
_ { } [ ] # ( ) < > % : ; . ? * + - / ^ & | ~ ! = , \ " ' "
```

Συμβολοσειρά (String) ως πίνακας χαρακτήρων

Για να θεωρηθεί ένας πίνακας χαρακτήρων συμβολοσειρά θα πρέπει μετά τον τελευταίο χαρακτήρα του να υπάρχει ο χαρακτήρας με ascii κωδικό 0, ο οποίος είναι γνωστός ως **χαρακτήρας null** ('\0').

```
char name[20] = { 'h', 'e', 'l', 'l', 'o', '\0' };
```

```
char name[] = { 'h', 'e', 'l', 'l', 'o', '\0' };  
char name[] = "hello";
```

Δεσμεύουν χώρο για τον αριθμό των χαρακτήρων συν ένα για τον χαρακτήρα '\0'. Στην περίπτωση μας 6 (5+1).

Αναφορά
στοιχείου

Μνήμη

Διεύθυνση

name[0]

h

F000

name[1]

e

F001

name[2]

l

F002

l

F003

o

F004

\0

F005

F006

....

name[19]

F019

Το name είναι η διεύθυνση στο πρώτο στοιχείο του πίνακα

Είσοδος και έξοδος συμβολοσειράς

- Για **έξοδο συμβολοσειράς** χρησιμοποιούμε τη γενική συνάρτηση εξόδου **printf** με τον **προσδιοριστή %s**.
- Η συνάρτηση τυπώνει στην κύρια έξοδο τους χαρακτήρες της συμβολοσειράς μέχρι να συναντήσει τον χαρακτήρα null.

```
char message[20] = {'h','e','l','l','o','\0'};  
printf("message is %s\n", message);
```

Για **είσοδο συμβολοσειράς** δίνουμε συναρτήσεις :

scanf	(της τυπικής βιβλιοθήκης)
gets	(της τυπικής βιβλιοθήκης)
fgets	

```
scanf("%s", message);
```

Η συνάρτηση τοποθετεί μετά τον τελευταίο χαρακτήρα τον χαρακτήρα null που σηματοδοτεί το τέλος της συμβολοσειράς.

Συναρτήσεις Αλφαριθμητικών

▣ <stdio.h>

```
char *gets(char *s)    // διαβάζει string από το stdin στο s
                        // και επιστρέφει το s
```

```
int puts(const char *s) // γράφει το string s στο stdout
```

```
Char * fgets(char*s, int n, FILE *fstream); // διαβάζει το πολύ n-1
//χαρακτήρες σταματώντας αν συναντήσει χαρακτήρα νέας γραμμής
```

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

```
int main()
```

```
{
```

```
    char x[20];
```

```
    printf("Give: ");
```

```
    scanf("%19[^\n]s", x);
```

```
    puts(x);
```

```
}
```

<https://learn.microsoft.com/en-us/cpp/c-runtime-library/scanf-width-specification?view=msvc-170>

Hands-on (StringHandling)

Δράση 6-4 StringHandling

Αναπτύξτε το πρόγραμμα που ικανοποιεί τις απαιτήσεις της Δραστηριότητας StringHandling.

Δοκιμάστε και τις τρεις συναρτήσεις εισόδου (scanf, getStringV1, getStringV2).



Αναπτύξτε το πρόγραμμα και ελέγξτε τη συμπεριφορά και των τριών συναρτήσεων δίνοντας περισσότερους χαρακτήρες από αυτούς που δεσμεύσατε για το όνομα

Αντιστοίχιση σημειογραφίας πίνακα με σημειογραφία Δείκτη

Η C θεωρεί το όνομα του πίνακα ως δείκτη που δείχνει το πρώτο στοιχείο του πίνακα. Έτσι αν

```
char name[12] = "klea";
```

και

```
char *namePtr = name;
```

η πρόταση

```
printf("%c\n", *namePtr);
```

τυπώνει το πρώτο στοιχείο του πίνακα, και είναι ισοδύναμη με την

```
printf("%c\n", name[0]);
```

Η πρόταση δηλώνει τη μεταβλητή namePtr ως δείκτη σε χαρακτήρα και την βάζει να δείχνει στον πρώτο χαρακτήρα της συμβολοσειράς name

Αντιστοίχιση σημειογραφίας πίνακα με σημειογραφία Δείκτη

```
for (int i=0; name[i] != 0; i++) {  
    printf("\t\t%c\n", *namePtr);  
    namePtr++; // name++ not valid  
}
```

name[i] ↔ *(name+i)
&name[i] ↔ name+i

```
for (int i=0; name[i] != 0; i++)  
    printf("\t\t%c\n", *(name+i) );
```

Συναρτήσεις διαχείρισης συμβολοσειρών

```
int strlen(char str[]);  
char *strcat(char *string1, const char *string2);  
int strcmp(const char *string1, const char *string2);  
char *strcpy(char *string1, const char *string2);  
char *strncat(char *s1, const char *s2, size_t count);  
char *strncpy(char *s1, const char *s2, size_t count);
```

Hands-on (UsingStringhFunctions)

Δράση 6.5 UsingStringhFunctions

Αναπτύξτε ένα πρόγραμμα που θα επιδεικνύει τη χρήση συναρτήσεων της `string.h`. Ξεκινήστε από αυτές των οποίων έχουμε ήδη παραθέσει τις δηλώσεις των.



**Αναπτύξτε το πρόγραμμα και
χρησιμοποιήστε τον τελεστή `sizeof`.
Αξιοποιήστε τον τύπο `size_t`.**

Οργάνωση Διάλεξης

- Η δομή της «Μηχανής»
 - Von Neumann Architecture, Γλώσσα Μηχανής
- Δραστηριότητα 6.1 - AsciiCode
 - Ο τύπος χαρακτήρα (char) και ο ASCII κώδικας
 - Μονάδες αποθήκευσης, Διευθύνσεις και Δείκτες
- Μονάδες αποθήκευσης, Διευθύνσεις και Δείκτες
- Δραστηριότητα 6.2 - StringHandling
 - Συμβολοσειρά (String) και αλφαριθμητικό
 - Αντιστοίχιση σημειογραφίας πίνακα με σημειογραφία Δείκτη
 - Συναρτήσεις διαχείρισης συμβολοσειρών
- Δραστηριότητα 6.3 - GetStringVersions
 - Ο Alex, η getStringV2 και το chatbot

Δραστηριότητα 6.3 - GetStringVersions

Αναπτύξτε ένα πρόγραμμα που θα ορίζει και θα επιδεικνύει τη χρήση των δύο εκδόσεων της **getString** με τις δηλώσεις που δίνονται παρακάτω:

getString Version 1

```
void getStringV1(char message[], char str[]);
```

getString Version 2

Η δήλωση της getStringV2 με σημειογραφία πίνακα:

```
void getStringV2(char message[], char str[], int strSize);
```

Η δήλωση της getStringV2 με σημειογραφία δείκτη

```
void getStringV2(char *message, char *str, int strSize);
```

Hands-on (GetStringVersions)

Δράση 6.8 Ορισμός `getStringV2`

Αξιοποιήστε τη `getchar` για να δώσετε έναν ορισμό της `getStringV2()`.

Hands-on (getStringV2 με την getchar())

```
mainTestGetchar.c X
1  #include <stdio.h>
2  #include <stdlib.h>
3  char buf[6];
4
5  int main() {
6      char ch;
7      int i=0;
8      printf("TestGetchar \n");
9      printf("Enter a string:");
10     while((ch = getchar()) != '\n') {
11         buf[i++] = ch;
12         putchar(ch);
13     }
14     buf[i] = '\0';
15     printf("\nbuf:%s\n", buf);
16 }
```

Προσδιορίστε την έξοδο
του προγράμματος για
είσοδο 12345678



Hands-on (GetStringVersions)

Δράση 6-6 Ορισμός `getStringV1`

Αναπτύξτε σε πρώτη φάση την `getStringV1` και ελέγξτε τη συμπεριφορά του προγράμματος για είσοδο μεγαλύτερη από το χώρο που έχει δεσμευθεί για τη μεταβλητή `str`.



**Αναπτύξτε το πρόγραμμα και
ελέγξτε τη συμπεριφορά της
`getStringV1`**