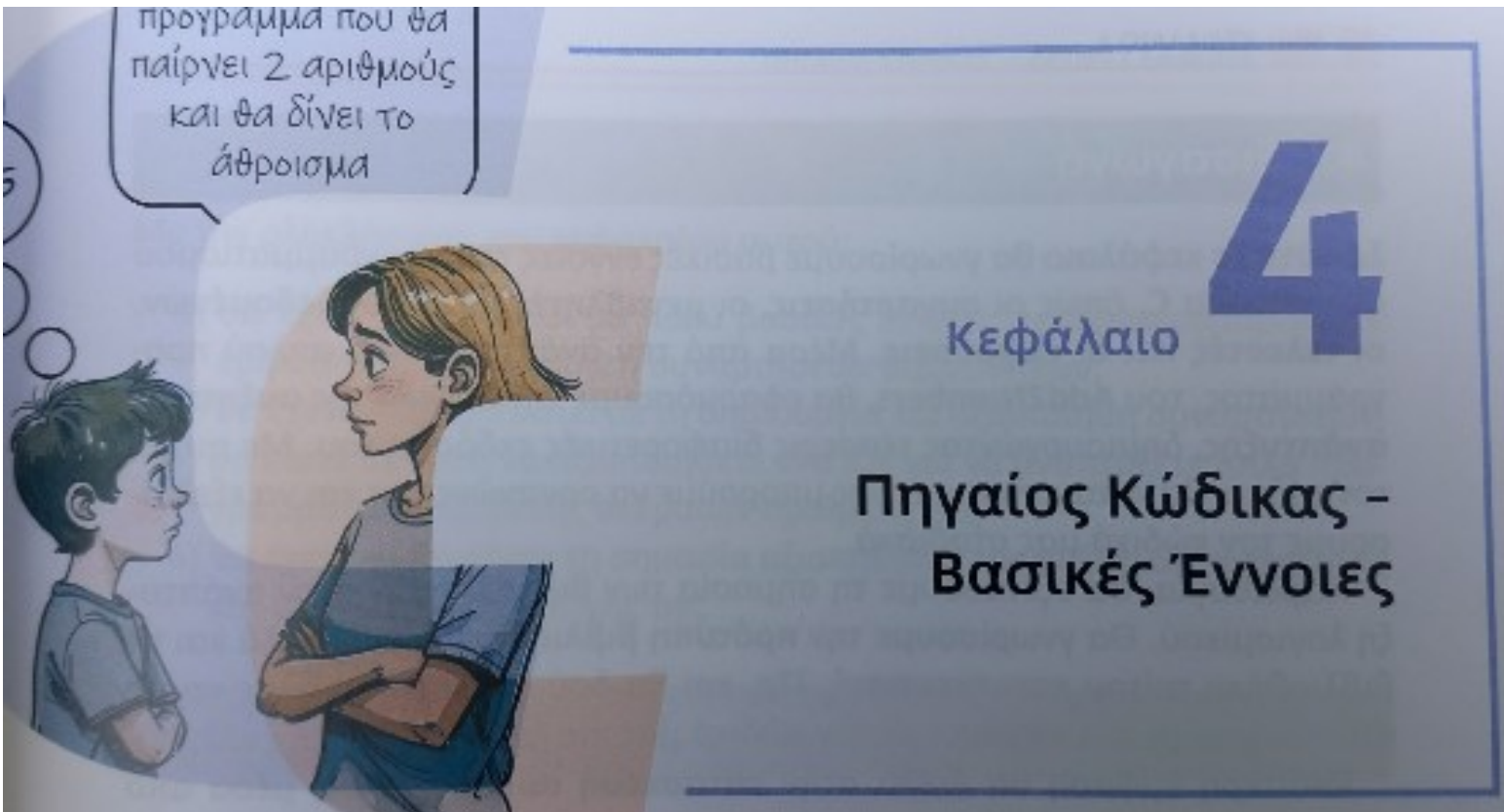


# Εισαγωγή στον Προγραμματισμό (Introduction to Programming)

(CEID\_NY131)

## Πηγαίος Κώδικας – Βασικές Έννοιες

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



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

Source Code  
Basic Concepts

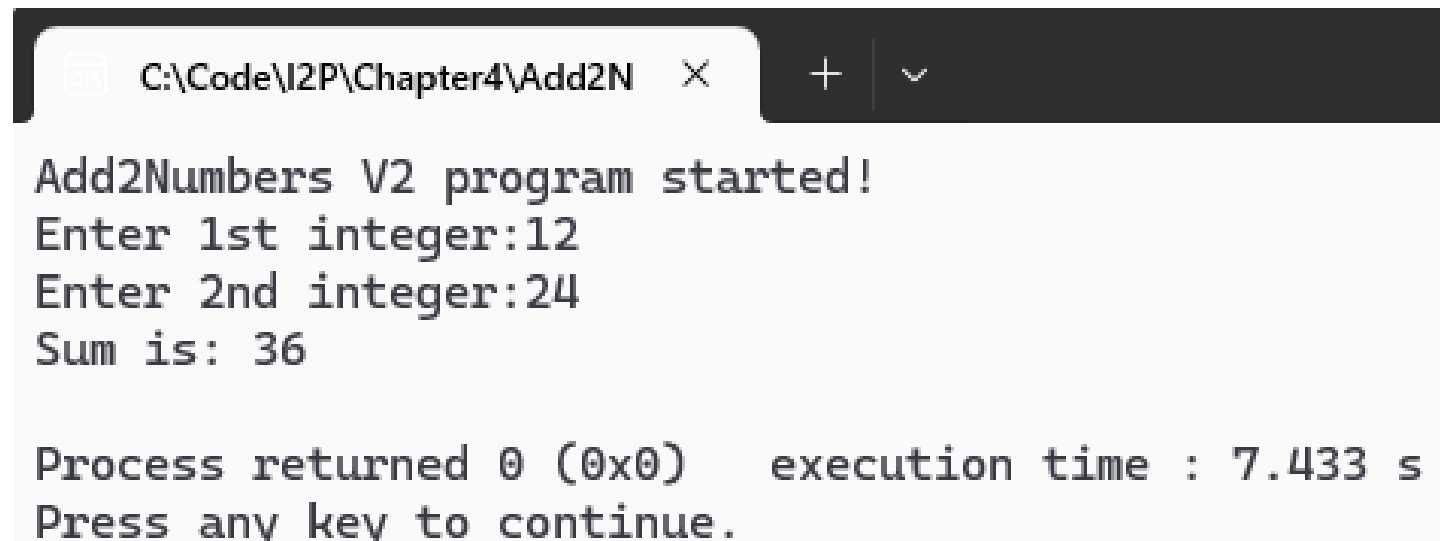
# Στόχος Ενότητας

- **εισάγει** βασικές έννοιες πηγαίου κώδικα και το πως η C τις υποστηρίζει
  - **συναρτήσεις, μεταβλητές, τύποι δεδομένων, τελεστές και εκφράσεις.**
- Εισάγει την έννοια της βιβλιοθήκης στον προγραμματισμό
  - παρουσιάζει τον τρόπο αξιοποίησης βιβλιοθηκών (τυπικής βιβλιοθήκης και βιβλιοθήκης τρίτου κατασκευαστή) στην διαδικασία ανάπτυξης προγράμματος.
- **προτείνει** την αξιοποίηση της τεχνικής της αυξητικής ανάπτυξης (incremental development)
  - Αναπτύσσουμε ένα απλό πρόγραμμα, το Add2Numbers, δημιουργώντας τέσσερις διαφορετικές εκδόσεις του, για να κατανοήσουμε πώς μπορούμε να οργανώνουμε και να εξελίσσουμε τον κώδικά μας σταδιακά.
- **Εισάγει** την δημιουργία συναρτήσεων ως μηχανισμό επαναχρησιμοποίησης κώδικα

# Δραστηριότητα 4-1 Add2Numbers

**Αναθέστε στη μηχανή το έργο** της πρόσθεσης δύο ακέραιων αριθμών και της εμφάνισης του αποτελέσματος στην οθόνη.

Σχόλιο: Εκτελώντας τη διεργασία *Add2Numbers* η μηχανή προσφέρει στο χρήστη μια **υπηρεσία**, την οποία αυτός μπορεί να αξιοποιήσει για να βρει το άθροισμα δύο αριθμών. Στα πλαίσια αυτά θα πρέπει να δώσει στη μηχανή τους δύο αριθμούς, η μηχανή να τους πάρει, να τους προσθέσει και να δώσει το αποτέλεσμα στο χρήστη.



```
C:\Code\I2P\Chapter4\Add2N  X  +  v

Add2Numbers V2 program started!
Enter 1st integer:12
Enter 2nd integer:24
Sum is: 36

Process returned 0 (0x0)    execution time : 7.433 s
Press any key to continue.
```

# Βήματα Διαδικασίας Ανάπτυξης Προγράμματος

Βήμα 1	Γραφική αναπαράσταση διεργασίας
Βήμα 2	Λεκτική Περιγραφή
Βήμα 3	Μετατροπή Λεκτικής περιγραφής σε πηγαίο κώδικα. Δημιουργία project
Βήμα 4	Μετασχηματισμός πηγαίου κώδικα σε εκτελέσιμο (Build)
Βήμα 5	Εκτέλεση προγράμματος (Run), έλεγχος και διορθώσεις αν απαιτούνται

# Συνοπτική περιγραφή διαδικασίας εκτέλεσης Δραστηριότητας

1	Δημιουργία Λεκτικής Περιγραφής
2	Μετατροπή Λεκτικής Περιγραφής σε Πηγαίο Κώδικα
3	Δημιουργία 1ης έκδοσης του Add2Numbers
4	Δημιουργία 2ης έκδοσης του Add2Numbers
5	Δημιουργία 3ης έκδοσης του Add2Numbers
6	Δημιουργία 4ης έκδοσης του Add2Numbers

# Δημιουργία Λεκτικής Περιγραφής

Δημιουργούμε μια **ακολουθία προτάσεων**, σε **προστακτική μορφή**, οι οποίες προσδιορίζουν τις ενέργειες που πρέπει να κάνει μια οντότητα, πιθανόν άνθρωπος, και **τη σειρά** με την οποία πρέπει να τις εκτελέσει για να φέρει σε πέρας το έργο.

**Δεν απαιτούνται γνώσεις προγραμματισμού υπολογιστών.**

## Λεκτική περιγραφή για το Add2Numbers

1. Πάρε τον πρώτο αριθμό
2. Πάρε τον δεύτερο αριθμό
3. Πρόσθεσε τους δύο αριθμούς
4. Εμφάνισε το άθροισμα στην οθόνη

Προστακτική!  
Περιγράφει ενέργειες που πρέπει να εκτελεστούν

# Προϋποθέσεις για ανάθεση του έργου στη μηχανή

Η Μηχανή θα πρέπει να μπορεί να εκτελέσει τις διεργασίες που η Λεκτική Περιγραφή αναφέρει.

## Λεκτική Περιγραφή (Add2Numbers)

1. Πάρε τον πρώτο αριθμό
2. Πάρε τον δεύτερο αριθμό
3. Πρόσθεσε τους δύο αριθμούς
4. Εμφάνισε το άθροισμα στην οθόνη

Βασικές Διεργασίες που η μηχανή πρέπει να υποστηρίζει

- Διεργασία εισόδου αριθμού (Input)
- Δυνατότητα αποθήκευσης αριθμού (storage-memory)
- Διεργασία πρόσθεσης αριθμών
- Διεργασία εμφάνισης αριθμού στην οθόνη

Η **σύνθεση** αυτών των διεργασιών είναι το πηγαίο πρόγραμμα το οποίο θα προκύψει με την μετατροπή της Λεκτικής περιγραφής σε πηγαίο κώδικα.

# Μετατροπή Λεκτικής περιγραφής σε πηγαίο κώδικα

- Διεργασίες και Συναρτήσεις
- Όνομα και σημασία συνάρτησης
- Η συνάρτηση getInt

## Λεκτική Περιγραφή (Add2Numbers)

1. Πάρε τον πρώτο αριθμό
2. ...

## Πηγαίος Κώδικας (Add2Numbers)

1. `getInt("Enter 1st integer:");`
2. ...



# Συνάρτηση, κλήση συνάρτησης και ορίσματα

**Η συνάρτηση (function) είναι** μια κατασκευή της γλώσσας προγραμματισμού που επιτρέπει στον προγραμματιστή να περιγράψει τις ενέργειες που πρέπει να κάνει η μηχανή για να εκτελέσει επιτυχώς μια συγκεκριμένη διεργασία.

**Η συνάρτηση `getInt`** περιγράφει τις ενέργειες που πρέπει να κάνει η μηχανή για να διαβάσει από το πληκτρολόγιο έναν αριθμό που θα πληκτρολογήσει ο χρήστης.

**Ποιες είναι οι ενέργειες αυτές;**

## Πηγαίος Κώδικας (`Add2Numbers`)

1. `getInt("Enter 1st integer:");`
2. ...

Κλήση της συνάρτησης `getInt`

Όρισμα συνάρτησης.  
Κάνει τη συνάρτηση επαναχρησιμοποιήσιμη

- Μνήμη υπολογιστή – Διεύθυνση θέσεων μνήμης
- Μεταβλητή

## ■ Η μεταβλητή

- **δίνει τη δυνατότητα** στον προγραμματιστή να αναφερθεί σε μία θέση μνήμης του υπολογιστή και να τον προστάξει να αποθηκεύσει εκεί κάποια πληροφορία, αλλά και να αναφερθεί στην πληροφορία αυτή για να αναθέσει στη μηχανή τη διαχείριση της.
- **είναι** μία λέξη που ορίζουμε εμείς ως προγραμματιστές και πρέπει να προσδιορίζει την πληροφορία στην οποία αυτή αναφέρεται.

### Πηγαίος Κώδικας (Add2Numbers)

1. `num1 = getInt("Enter 1st integer:");`
2. `num2 = getInt("Enter 2nd integer:");`

# Τύπος Δεδομένων

Έχετε την ικανότητα να διακρίνετε αν μια πληροφορία είναι αριθμός ή συμβολοσειρά. Την ικανότητα αυτή δεν την έχει η μηχανή (τουλάχιστο όσον αφορά την C), οπότε θα πρέπει ο προγραμματιστής να φροντίσει να την ενημερώσει για τον τύπο της πληροφορίας που πρόκειται να αποθηκευτεί στην θέση μνήμης που αναφέρεται μία μεταβλητή. Αυτός είναι **ο ρόλος του τύπου δεδομένων** για μια γλώσσα προγραμματισμού.

## Βασικοί τύποι δεδομένων της C

Για να υποστηρίξει τη **διαχείριση πληροφορίας** η C ορίζει ορισμένους βασικούς τύπους, μεταξύ των οποίων:

- ο τύπος του ακεραίου (που συμβολίζεται με τη λέξη **int**),
- οι δύο τύποι για αριθμούς με δεκαδικά (**float** και **double**), και,
- ο τύπος του χαρακτήρα (που συμβολίζεται με την λέξη **char**).

# Πρόταση Δήλωσης μεταβλητής

5

```
6 int num1;    // protasi dilwsis genikis (global) metavlitis
```

Η γραμμή 6 είναι μια πρόταση δήλωσης γενικής (global) μεταβλητής. Ζητά από το σύστημα να δεσμεύσει χώρο στη μνήμη για την αποθήκευση ενός ακέραιου αριθμού (**int**) και αποδίδει στη θέση αυτή το συμβολικό όνομα **num1** (όνομα μεταβλητής).

- **Μεταβλητή είναι ένα Συμβολικό όνομα**
  - που δίνει λύση στο πρόβλημα της αναφοράς σε πληροφορία της κύριας μνήμης του υπολογιστή.
- Το **όνομα της μεταβλητής** μας επιτρέπει να αναφερόμαστε
  - **στη τιμή**, χωρίς να είναι απαραίτητο να γνωρίζουμε την ακριβή διεύθυνση της μνήμης που αυτή είναι αποθηκευμένη,
  - **στην θέση της μνήμης** για να τροποποιήσουμε την τρέχουσα-τιμή της.

**count = count + 1**

# Δήλωση του τύπου της μεταβλητής

5

```
6 int num1;    // protasi dilwsis genikis (global) metavlitis
```

Αν θέλουμε να αποθηκεύσουμε δεδομένα πρέπει **πρώτα να δηλώσουμε** την κατάλληλη μεταβλητή. Η δήλωση μεταβλητής **προσδιορίζει** εκτός από το όνομα της **και τον τύπο του δεδομένου** για το οποίο αυτή δηλώνεται.

- Οι σύγχρονες προστακτικές γλώσσες
  - **περιλαμβάνουν** ορισμένους **ενσωματωμένους τύπους**,
    - **int** (ακέραιος), **char** (χαρακτήρας), **float**, **double** ...
  - **προσφέρουν** ένα **μηχανισμό για τον ορισμό νέων τύπων**
    - `struct student { ... };`
    - `struct rectangle { ... };`

Τα βασικά για τύπους δεδομένων θα βρείτε στο [Module13-Variables&DataTypes.pdf](#) (slides 9-14)

# Μεταβλητή – Δήλωση και απόδοση τιμής

## Δήλωση

**int num1;**

Η πρόταση δηλώνει μία μεταβλητή με όνομα num1 και δεσμεύει χώρο για αποθήκευση ακεραίου

## Δήλωση και απόδοση τιμής στο χρόνο συγγραφής (edit-time)

**int num1 = 12;**

Η πρόταση δηλώνει μία μεταβλητή με όνομα num1, δεσμεύει χώρο για αποθήκευση ακεραίου και αποδίδει αρχική τιμή 12

## Απόδοση τιμής στο χρόνο εκτέλεσης (run-time)

**num1 = getInt("Enter 1st integer:");**

Η πρόταση αποδίδει στην μεταβλητή num1, την τιμή που θα διαβάσει η συνάρτηση getInt() από την βασική είσοδο.

# Αυξητική Ανάπτυξη (incremental development)

«A complex system that works is invariably found to have evolved from a simple system that worked...

A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over, beginning with a working simple system»

<J. Gall>

**Αναπτύξτε το πρόγραμμά σας σταδιακά σε διαδοχικές εκδόσεις με κάθε έκδοση να προσθέτει επιπλέον λειτουργικότητα στην προηγούμενη που έχει ήδη ολοκληρωθεί.**

# Add2Numbers - Αυξητική Ανάπτυξη

1η Έκδοση: Υλοποιεί τη λειτουργικότητα των 2 πρώτων προτάσεων της ΛΠ

## Λεκτική Περιγραφή (Add2Numbers)

1. Πάρε τον πρώτο αριθμό
2. Πάρε τον δεύτερο αριθμό
3. Πρόσθεσε τους δύο αριθμούς
4. Εμφάνισε το άθροισμα στην οθόνη

2η Έκδοση: Υλοποιεί όλη τη λειτουργικότητα (4 προτάσεις της ΛΠ)



# 1<sup>η</sup> έκδοση του Add2Numbers

The screenshot displays a C++ IDE interface. On the left, the 'Management' pane shows a workspace named 'Add2Numbers' containing a 'Sources' folder with the file 'main.c'. The main editor window, titled 'main.c', contains the following code:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6      printf("Add2Numbers program started!\n");
7      num1 = getInt("Enter 1st integer:");
8      num2 = getInt("Enter 2nd integer:");
9      return 0;
10 }
```

Below the editor, the 'Logs & others' pane shows the build output from the GNU GCC Compiler. The output indicates two errors and one warning:

File	Line	Message
=== Build: Release in Add2Numbers (compiler: GNU GCC Compiler) ===		
In function 'main':		
C:\Code\I2P\A...	7	error: 'num1' undeclared (first use in this function)
C:\Code\I2P\A...	7	note: each undeclared identifier is reported only once for each function it appears in
C:\Code\I2P\A...	7	warning: implicit declaration of function 'getInt'; did you mean 'getenv'? [-Wimplicit...
C:\Code\I2P\A...	8	error: 'num2' undeclared (first use in this function)
=== Build failed: 2 error(s), 1 warning(s) (0 minute(s), 0 second(s)) ===		

# Βιβλιοθήκη (library)

Ο όρος βιβλιοθήκη χρησιμοποιείται στον προγραμματισμό για να αναφερθούμε σε ένα σύνολο συναρτήσεων που υλοποιούν διεργασίες και οι οποίες είναι σε μορφή που μπορούν να αξιοποιηθούν στα προγράμματα μας χωρίς να χρειάζεται να «ανακαλύψουμε πάλι τον τροχό».

- Η **printf** είναι μια από τις συναρτήσεις που μας δίνει μια ειδική βιβλιοθήκη, που ονομάζεται **πρότυπη βιβλιοθήκη** της C (standard C library).
- Αντίθετα η συνάρτηση **getInt** δεν βρίσκεται στην πρότυπη βιβλιοθήκη αλλά σε μία άλλη την οποία καλούμε **βιβλιοθήκη τρίτου κατασκευαστή** (third party library).

# Πρότυπη βιβλιοθήκη της C (standard C library)

- ορίζει ένα σύνολο από συναρτήσεις που υλοποιούν βασικές διεργασίες.
- η επαναχρησιμοποίηση έτοιμων τμημάτων κώδικα μειώνει σημαντικά το χρόνο ανάπτυξης, αυξάνοντας ταυτόχρονα την αξιοπιστία.

Συνάρτηση		
printf	stdio.h	int printf(const char *format-string, arg-list);
putc <sup>1</sup>	stdio.h	int putc(int c, FILE *stream);
putchar <sup>1</sup>	stdio.h	int putchar(int c);
putenv	stdlib.h	int *putenv(const char *varname);
puts	stdio.h	int puts(const char *string);
putwc <sup>6</sup>	stdio.h wchar.h	wint_t putwchar(wchar_t wc, FILE *stream);

Αρχείο επικεφαλίδας  
(header file)

# Βιβλιοθήκη τρίτου κατασκευαστή – i2p library

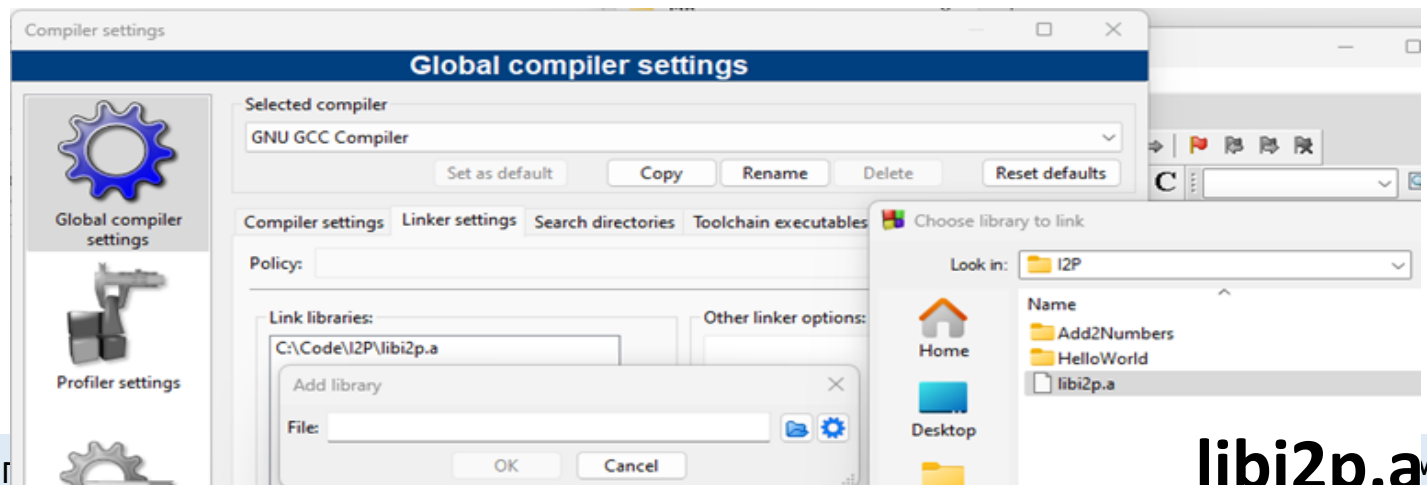
- Υλοποιεί διεργασίες που συχνά συναντώνται σε κατηγορίες προγραμμάτων.
- Η βιβλιοθήκη i2p (introduction to programming) δημιουργήθηκε για να διευκολύνει την εισαγωγή στην τέχνη του προγραμματισμού με την γλώσσα C.
- Αποτελείται από 2 αρχεία:
  - Το **libi2p.a** περιέχει τον εκτελέσιμο κώδικα των συναρτήσεων που η βιβλιοθήκη περιλαμβάνει.
  - Το **i2p.h** είναι το αρχείο επικεφαλίδας της βιβλιοθήκης.

## Ενέργειες για αξιοποίηση της βιβλιοθήκης i2p

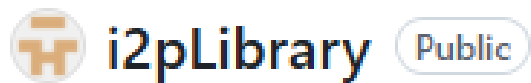
<https://sites.google.com/view/i2art-of-programming/i2p-library>

```
#include "..\i2p.h
```

```
#include "C:\Code\I2P\i2p.h"
```



# i2p library στο github



main

1 Branch 0 Tags

Go to file

ThramboulidisKleanthis Add files via upload

LICENSE Initial commit

README.md Initial commit

i2p.h Add files via upload

libi2p.a Add files via upload

# Hands-on (Add2Numbers Version 1)



**Αναπτύξτε την 1<sup>η</sup> έκδοση του προγράμματος Add2Numbers αξιοποιώντας την i2p library**

**Ελέγξτε αν αποδόθηκαν σωστά οι τιμές στις μεταβλητές num1 και num2**

# Δράση 4.3 – 2<sup>η</sup> έκδοση του Add2Numbers

1<sup>η</sup> Έκδοση: Υλοποιεί τη λειτουργικότητα των 2 πρώτων προτάσεων της ΛΠ

## Λεκτική Περιγραφή (Add2Numbers)

1. Πάρε τον πρώτο αριθμό
2. Πάρε τον δεύτερο αριθμό
3. Πρόσθεσε τους δύο αριθμούς
4. Εμφάνισε το άθροισμα στην οθόνη

2<sup>η</sup> Έκδοση: Υλοποιεί όλη τη λειτουργικότητα (4 προτάσεις της ΛΠ)

# Hands-on (Add2Numbers Version 2)



**Αναπτύξτε την 2<sup>η</sup> έκδοση του προγράμματος Add2Numbers αξιοποιώντας την 1<sup>η</sup>**

**Ελέγξτε αν υλοποιεί σωστά την απαιτούμενη λειτουργικότητα**



# Ορίζοντας τη δική μας συνάρτηση (Add2NumbersV3)

## Δομή προγράμματος C

mainUsingSumV1.c X

```
1 #include <stdio.h>
2 #include "../i2p.h"
```

Προτάσεις Προεπεξεργαστή

```
4 void sum(void);
```

Πρόταση δήλωσης της sum

```
6 int num1, num2;
7 int result;
```

Προτάσεις δήλωσης μεταβλητών

```
9 int main() {
10     printf("Add2Numbers program using sumV1 started!\n");
11     num1 = getInt("Enter 1st integer:");
12     num2 = getInt("Enter 2nd integer:");
13     sum();
14     displayInt("Sum is:", result);
15     return 0;
16 }
```

Κλήση της συνάρτησης sum  
(function call)

Ορισμός της συνάρτησης main

```
18 void sum(void) {
19     result = num1 + num2;
20 }
```

Ορισμός της συνάρτησης sum

# Hands-on (Add2Numbers Version 3)

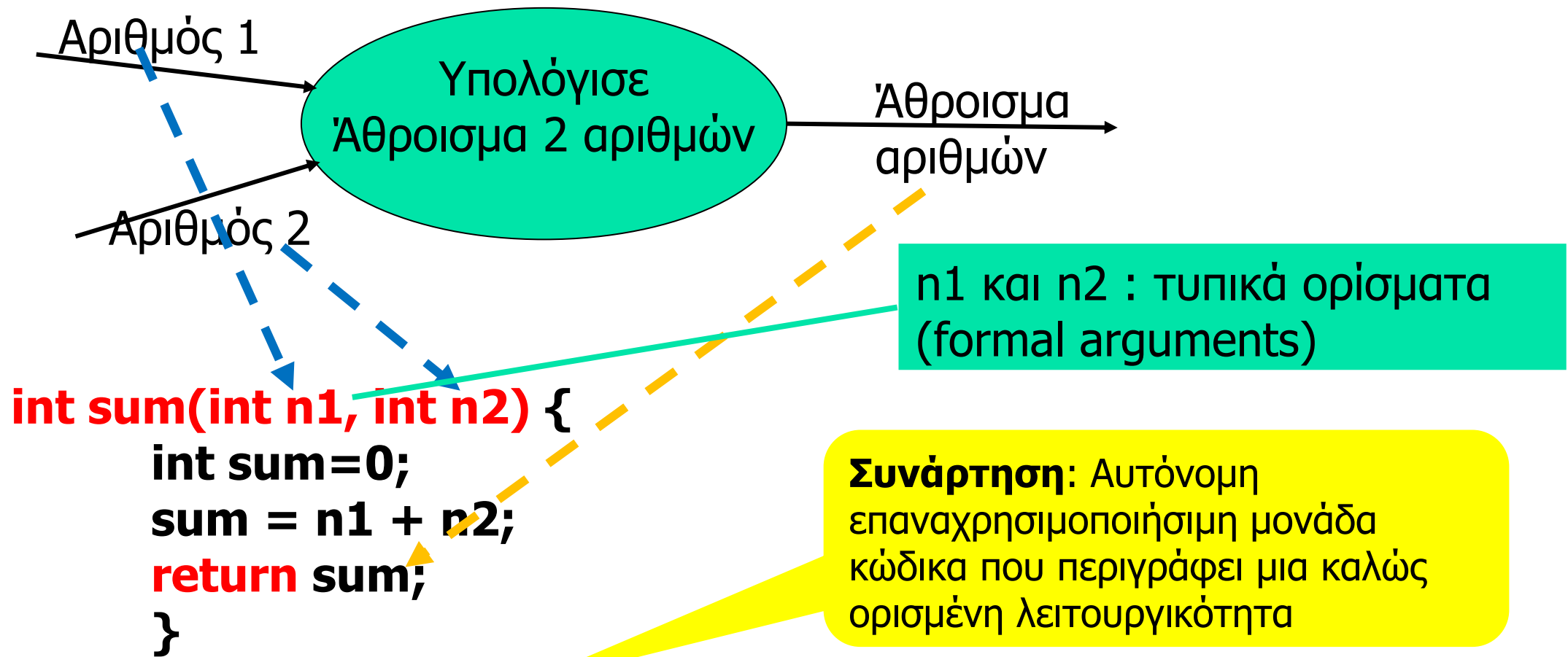


**Αναπτύξτε την 3<sup>η</sup> έκδοση του προγράμματος Add2Numbers ορίζοντας τη δική σας συνάρτηση sum**

**Ελέγξτε αν υλοποιεί σωστά την απαιτούμενη λειτουργικότητα**

# Διεργασία ως Reusable function - Η συνάρτηση sum

Η διεργασία sum ως αυτόνομη επαναχρησιμοποιήσιμη μονάδα λειτουργικότητας



# Η sum επαναχρησιμοποιήσιμη (Add2NumbersV4)

```
4  int sum(int n1, int n2);
5
6  int num1, num2;
7  int result;
8
9  int main() {
10     printf("Add2Numbers program using sumV2 started!\n");
11     num1 = getInt("Enter 1st integer:");
12     num2 = getInt("Enter 2nd integer:");
13     result = sum(num1, num2);
14     displayInt("Sum is:", result);
15     return 0;
16 }
17
18 int sum(int n1, int n2) {
19     int result;
20     result = n1 + n2;
21     return result;
22 }
```

Δήλωση της συνάρτησης sum

Τυπικά ορίσματα

Κλήση sum

Πραγματικά ορίσματα

Ορισμός της sum ως  
επαναχρησιμοποιήσιμη

Η sum δεν χρησιμοποιεί γενικές μεταβλητές οπότε είναι  
επαναχρησιμοποιήσιμη (reusable)

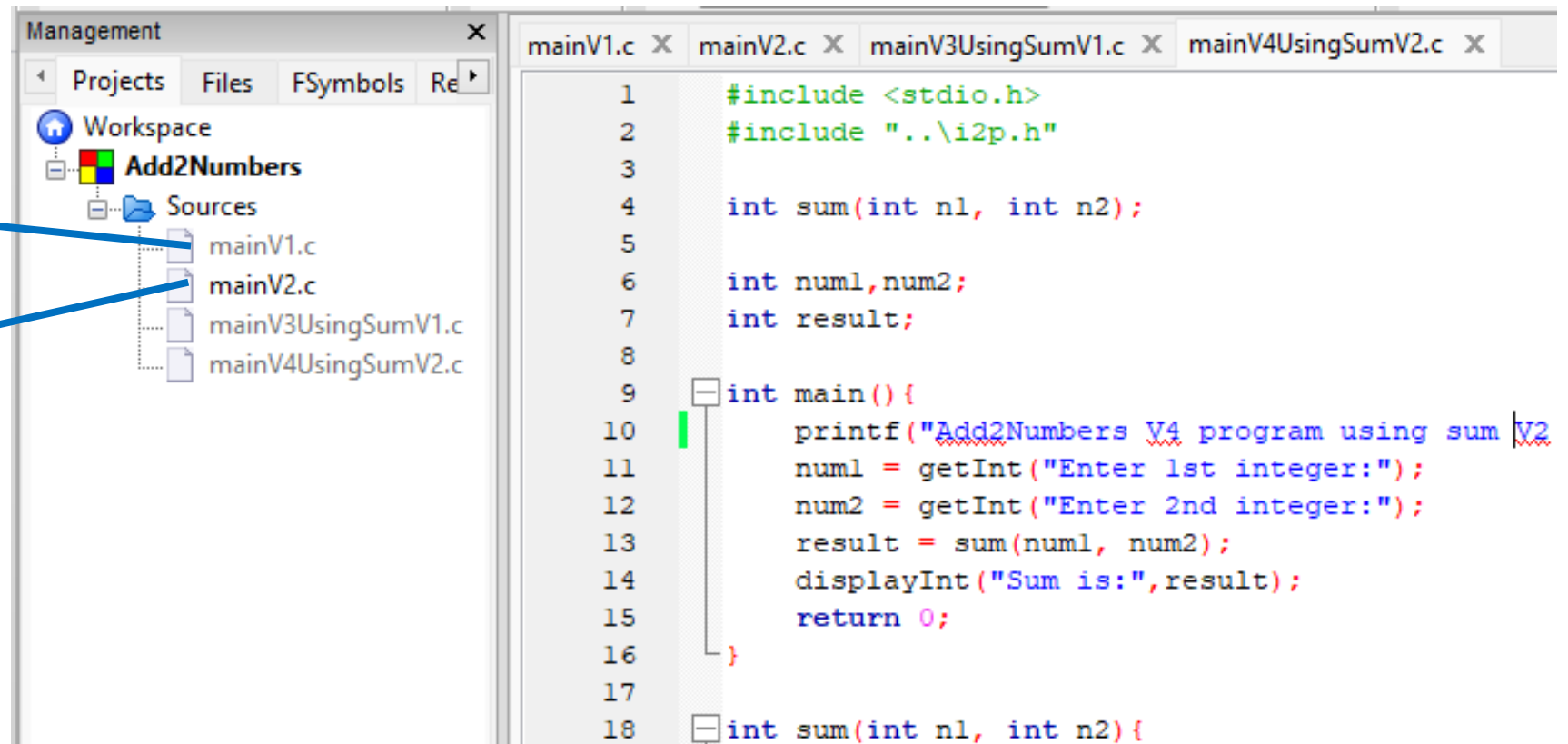
Επιστρεφόμενη τιμή συνάρτησης

# Διαχείριση εκδόσεων ενός προγράμματος

- Ένα πρόβλημα που έχουμε να αντιμετωπίσουμε όταν υιοθετούμε την αυξητική ανάπτυξη είναι αυτό της διαχείρισης των διαφορετικών εκδόσεων που αναπτύσσουμε.
- Οι εκδόσεις αυτές μπορεί να αφορούν προσθήκη επιπλέον λειτουργικότητας ή εναλλακτικές υλοποιήσεις.
- Στην φάση εκμάθησης προγραμματισμού, είναι καλή πρακτική να διατηρείτε τις εκδόσεις αυτές.

Options->Disable both

Options->Enable both



```
Management
├── Projects
├── Files
├── FSymbols
└── Re...

Workspace
├── Add2Numbers
│   └── Sources
│       ├── mainV1.c
│       ├── mainV2.c
│       ├── mainV3UsingSumV1.c
│       └── mainV4UsingSumV2.c
└── ...

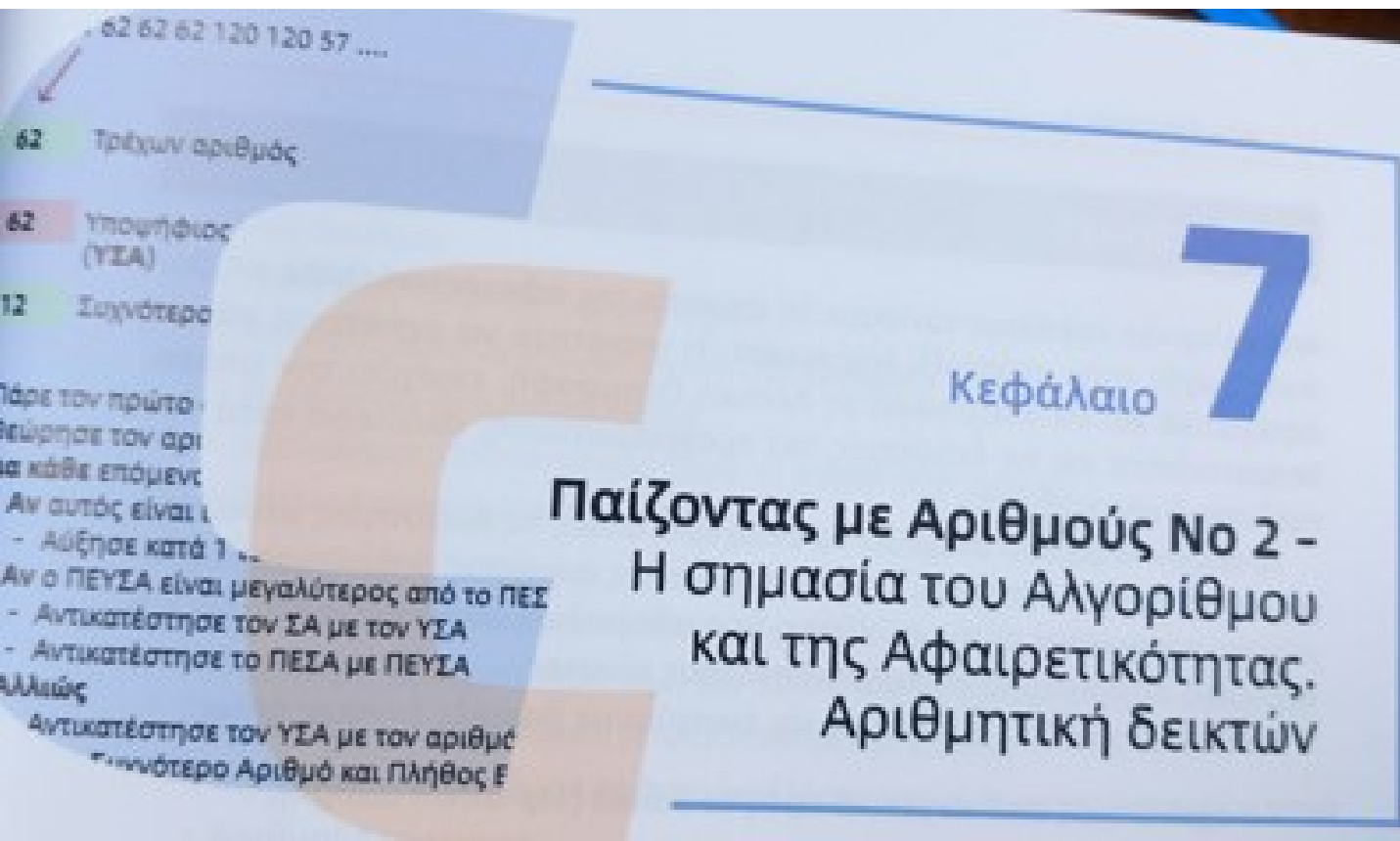
mainV1.c X  mainV2.c X  mainV3UsingSumV1.c X  mainV4UsingSumV2.c X

1  #include <stdio.h>
2  #include "../i2p.h"
3
4  int sum(int n1, int n2);
5
6  int num1, num2;
7  int result;
8
9  int main() {
10     printf("Add2Numbers V4 program using sum V2");
11     num1 = getInt("Enter 1st integer:");
12     num2 = getInt("Enter 2nd integer:");
13     result = sum(num1, num2);
14     displayInt("Sum is:", result);
15     return 0;
16 }
17
18 int sum(int n1, int n2) {
```

# 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.

# Η ενότητα

- **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.



# Δραστηριότητα 7.1 – Divisors

**Αναπτύξτε ένα πρόγραμμα** σύμφωνα με το οποίο το σύστημα θα δέχεται ως είσοδο ένα αριθμό και θα ελέγχει αν αυτός είναι πρώτος (prime). Αν ο αριθμός δεν είναι πρώτος, θα εμφανίζει τους διαιρέτες του και το πλήθος τους.

- Η Δραστηριότητα έχει στόχο την εξάσκηση με την **top-down τεχνική**, την τεχνική του **partitioning**, καθώς και με εναλλακτικούς αλγορίθμους με τη μορφή Λεκτικής Περιγραφής και τη μετατροπή τους σε πηγαίο κώδικα.

# Hands-on (Δράση 7-1 Αφαιρετική Λεκτική Περιγραφή)

**Αναπτύξτε ένα πρόγραμμα** σύμφωνα με το οποίο το σύστημα θα δέχεται ως είσοδο ένα αριθμό και θα ελέγχει αν αυτός είναι πρώτος (prime). Αν ο αριθμός δεν είναι πρώτος, θα εμφανίζει τους διαιρέτες του και το πλήθος τους.



**Δώστε μια αφαιρετική Λεκτική Περιγραφή χωρίς να ασχοληθείτε με το πως ελέγχουμε αν ένας αριθμός είναι πρώτος και πως βρίσκουμε τους διαιρέτες του. Δώστε τον αντίστοιχο πηγαίο κώδικα.**

# Δράση 7-1 Αφαιρετική Λεκτική Περιγραφή

## Λεκτική Περιγραφή 7-1 Divisors (1η έκδοχή)

1. Πάρε τον αριθμό
2. Αν ο **αριθμός είναι Πρώτος**
  - 2.1 Ενημέρωσε τον χρήστη για αυτό  
Αλλιώς
  - 2.2 **Εμφάνισε διαιρέτες του αριθμού**

Καταγράφει δύο βασικές διεργασίες.  
Μας επιτρέπει να αγνοήσουμε στο  
επίπεδο αυτό τις λεπτομέρειες

### ■ partitioning

- μας επιτρέπει να δουλέψουμε σταδιακά στην ανάπτυξη του προγράμματος εστιάζοντας κάθε φορά σε ένα επιμέρους τμήμα του

### ■ top-down

- ξεκινάμε από το γενικό και προχωράμε σταδιακά (αυξητική ανάπτυξη) προς το ειδικό

# Hands-on (Δράση 7-2 1η έκδοση - Δομή προγράμματος)

Δώστε τον πηγαίο κώδικα που αντιστοιχεί στη Λεκτική Περιγραφή





**Δώστε κατάλληλα ονόματα στις συναρτήσεις που αναπαριστούν τις βασικές διεργασίες που καταγράφει η Λεκτική Περιγραφή.**

**Στη φάση αυτή δεν ασχολούμαστε με τον ορισμό των συναρτήσεων.**

# Hands-on (Δράση 7-2 1η έκδοση - Δομή προγράμματος)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <stdbool.h>
4  #include "../..\i2p.h"
5
6  bool isPrime(int num);
7  void displayDivisors(int num);
8
9  int num;
10
11  int main(int argc, char *argv[]) {
12      printf("DivisorsV1\n");
13      num=getInt("Dose arithmo:");
14      if(isPrime(num))
15          printf("O arithmos %d einai protos\n", num);
16      else
17          displayDivisors(num);
18      return 0;
19  }
```

# Hands-on (Δράση 7-2 1η έκδοση - Δομή προγράμματος)

```
20
21  bool isPrime(int num) {
22     printf("isPrime executed for num =%d\n", num);
23     // return true;
24     return false;
25 }
26
27  void displayDivisors(int num) {
28     printf("displayDivisors executed for num =%d\n", num);
29 }
```

# Hands-on (Δράση 7-3 2<sup>η</sup> έκδοση – Αλγόριθμος προσδιορισμού αριθμού ως πρώτου)

Αναζητήστε εναλλακτικούς αλγορίθμους για τη Λεκτική Περιγραφή της διεργασίας ελέγχου αν ένας αριθμός είναι πρώτος. Δώστε τους αντίστοιχους πηγαίους κώδικες.



**Προχωράμε στον ορισμό της  
συνάρτησης isPrime αλλά ξεκινάμε  
από την Λεκτική Περιγραφή της.**

# Hands-on Αλγόριθμοι προσδιορισμού αριθμού ως πρώτου 1/2

“Υπάρχουν ποικίλες μέθοδοι για να προσδιορίσουμε αν ένας αριθμός  $n$  είναι πρώτος.

Η πιο βασική μέθοδος, η **δοκιμαστική διαίρεση**, έχει μικρή πρακτική χρησιμότητα επειδή είναι αργή.

Ένα τμήμα των σύγχρονων μεθόδων για τον προσδιορισμό αν ένας αριθμός είναι πρώτος είναι εφαρμόσιμο για όλους τους αριθμούς, ενώ οι πιο αποτελεσματικές μέθοδοι είναι διαθέσιμες μόνο για συγκεκριμένες κατηγορίες αριθμών.

Οι περισσότερες από αυτές τις μεθόδους λένε μόνο αν ο αριθμός είναι πρώτος ή όχι. Οι μέθοδοι, οι οποίες επιπλέον βρίσκουν και έναν ή περισσότερους παράγοντες του υπό εξέταση αριθμού ονομάζονται αλγόριθμοι παραγοντοποίησης.”

Πηγή: Wikipedia

## Αλγόριθμοι εύρεσης πρώτων – Περιγράφονται 4 αλγόριθμοι



## Αλγόριθμοι εύρεσης πρώτων

“Παρατίθενται μερικοί αλγόριθμοι (κατά σειρά ταχύτητας ή και απλότητας) για την εύρεση αν ο  $N \geq 2$  είναι πρώτος.

**Απλός 1** - Εξετάζουμε διαδοχικά όλους τους ακέραιους  $M < N$

**Απλός 2** - εξετάζοντας όλους τους αριθμούς  $M < N/2$

**Απλός 3** - εξετάζοντας όλους τους αριθμούς  $M$  που είναι μικρότεροι από την τετραγωνική ρίζα του  $N$ ,

**Απλός 4** - Εφαρμόζοντας το Θεώρημα του Ουίλσον.”

Πηγή: Wikipedia

# Hands-on – 1<sup>η</sup> έκδοση της isPrime (αλγόριθμος Απλός1)

1<sup>η</sup> έκδοση της isPrime (αλγόριθμος Απλός1).

```
21  bool isPrime(int num) {  
22      if (num < 2)  
23          return false;  
24      for (int i = 2; i * i <= num; i++)  
25          if (num % i == 0)  
26              return false;  
27      return true;  
28  }
```

# Hands-on – 2<sup>η</sup> έκδοση της isPrime (αλγόριθμος Απλός2)

```
21  bool isPrime(int num){ //Απλός 3
22      if (num < 2)
23          return false;
24      if (num == 2)
25          return true;
26      if (num % 2 == 0)
27          return false;
28      for (int i = 3; i <= sqrt(num); i += 2)
29          if (num % i == 0)
30              return false;
31      return true;
32  }
```

# Hands-on (Δράση 7-4 3<sup>η</sup> έκδοση – Αλγόριθμος εύρεσης των διαιρετών αριθμού)

Αναζητήστε εναλλακτικούς αλγορίθμους για τη Λεκτική Περιγραφή της διεργασίας εύρεσης των διαιρετών ενός αριθμού.  
Δώστε τους αντίστοιχους πηγαίους κώδικες.



**Προχωράμε στον ορισμό της  
συνάρτησης `displayDivisors` αλλά  
ξεκινάμε από τη Λεκτική Περιγραφή  
της.**

# Hands-on – 1<sup>η</sup> έκδοχή της displayDivisors

```
30 void displayDivisors(int num) {  
31     printf("Divisors of %d: ", num);  
32     for (int i = 1; i <= num; i++)  
33         if (num % i == 0)  
34             printf("%d ", i);  
35     printf("\n");  
36 }
```

# Hands-on – 2<sup>η</sup> εκδοχή της displayDivisors

```
35 void displayDivisors(int num){  
36     printf("Divisors of %d: ", num);  
37     for (int i = 1; i <= sqrt(num); i++)  
38         if (num % i == 0) {  
39             printf("%d ", i);  
40             if (i != num / i)  
41                 printf("%d ", num / i);  
42         }  
43     printf("\n");  
44 }
```

# Στόχος Ενότητας

- Στόχος του κεφαλαίου 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.

## Δραστηριότητα 7.3 – MinAndMax

**Αναπτύξτε ένα πρόγραμμα** σύμφωνα με το οποίο το σύστημα θα δέχεται, από τη βασική είσοδο, μία ακολουθία  $n$  αριθμών και θα εμφανίζει τον αριθμό με τη μικρότερη τιμή καθώς και αυτόν με τη μεγαλύτερη.

- Η Δραστηριότητα έχει στόχο την εξάσκηση με:
  - τον ορισμό αλγορίθμων με τη μορφή Λεκτικής περιγραφής, και,
  - την αξιοποίηση δεικτών σε εναλλακτικές υλοποιήσεις.



# Hands-on MinAndMax

(Δράση 7-8 Πρώτα βήματα της διαδικασίας ανάπτυξης)

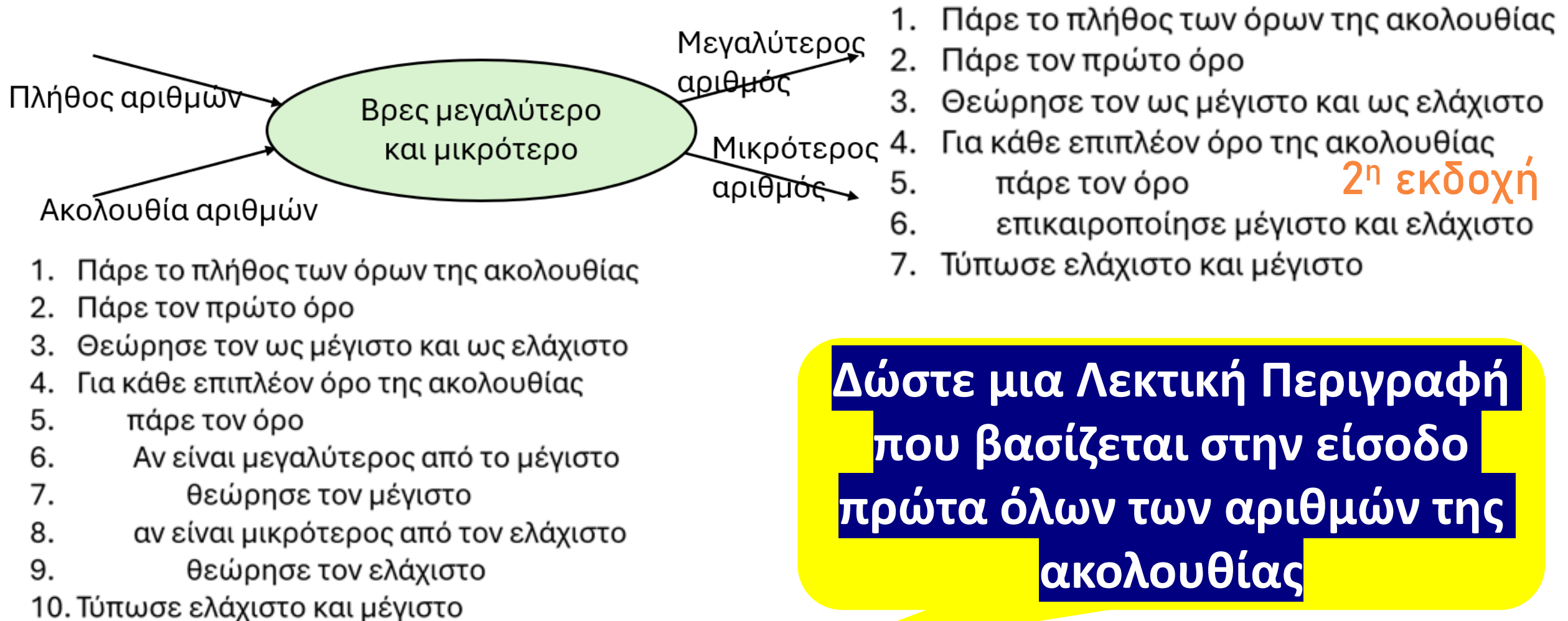
**Αναπτύξτε ένα πρόγραμμα** σύμφωνα με το οποίο το σύστημα θα δέχεται, από τη βασική είσοδο, μία ακολουθία  $n$  αριθμών και θα εμφανίζει τον αριθμό με τη μικρότερη τιμή καθώς και αυτόν με τη μεγαλύτερη.



Δώστε την **αφαιρετική αναπαράσταση** της διεργασίας και περιγράψτε σε μορφή λεκτικής περιγραφής τον **αλγόριθμο** που θα υλοποιεί το πρόγραμμα σας. Δώστε δύο τουλάχιστο αλγορίθμους. Αξιοποιήστε **αφαιρετικότητα** στις διεργασίες.

# MinAndMax (Δράση 7-8 Πρώτα βήματα της διαδικασίας ανάπτυξης)

**Αναπτύξτε ένα πρόγραμμα** σύμφωνα με το οποίο το σύστημα θα δέχεται, από τη βασική είσοδο, μία ακολουθία n αριθμών και θα εμφανίζει τον αριθμό με τη μικρότερη τιμή καθώς και αυτόν με τη μεγαλύτερη.



**Δώστε μια Λεκτική Περιγραφή  
που βασίζεται στην είσοδο  
πρώτα όλων των αριθμών της  
ακολουθίας**

# MinAndMax - Λεκτική Περιγραφή

## 3<sup>η</sup> εκδοχή

1. Πάρε το πλήθος των όρων της ακολουθίας
2. Πάρε την ακολουθία αριθμών
3. Θεώρησε τον πρώτο όρο ως μέγιστο και ως ελάχιστο
4. Για κάθε επόμενο όρο της ακολουθίας
5.     επικαιροποίησε μέγιστο και ελάχιστο
6. Τύπωσε ελάχιστο και μέγιστο

`getArrayOfInts`

## 4<sup>η</sup> εκδοχή

1. Πάρε το πλήθος των όρων της ακολουθίας
2. Πάρε την ακολουθία αριθμών
3. Ταξινόμησε την
4. Τύπωσε τον πρώτο και τον τελευταίο όρο της ταξινομημένης ακολουθίας

`sortIntArrayInc`