ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

ΑΝΟΙΚΤΑ ακαδημαϊκά μαθήματα ΠΠ

Προγραμματισμός Η/Υ

Ενότητα 1: Εισαγωγή στον Προγραμματισμό

Νίκος Καρακαπιλίδης, Καθηγητής
Δημήτρης Σαραβάνος, Καθηγητής
Πολυτεχνική Σχολή
Τμήμα Μηχανολόγων & Αεροναυπηγών Μηχανικών

Σκοποί ενότητας

- Απόκτηση δεξιοτήτων στην επίλυση προβλημάτων με διαδικαστικό τρόπο (μέσω προγραμματισμού)
- Ανάπτυξη αλγοριθμικής σκέψης
- Θεμελίωση βασικών αρχών προγραμματισμού, αλγοριθμικών τεχνικών και δομών δεδομένων
- Σχεδίαση, υλοποίηση, εκτέλεση και αποσφαλμάτωση προγραμμάτων, καθώς και αξιολόγηση εναλλακτικών λύσεων
- Εκμάθηση μιας υψηλού επιπέδου γλώσσας προγραμματισμού (C)

Περιεχόμενα ενότητας

- Συστήματα Λογισμικού
- Στυλ και Γλώσσες Προγραμματισμού

Μέρος 1ο

Συστήματα Λογισμικού

Συστήματα Λογισμικού

Ανάπτυξη Συστημάτων Λογισμικού

Ανάπτυξη Συστημάτων Λογισμικού

Οι Η/Υ είναι σήμερα παντού

- 1960: ~ 100 Η/Υ
- 2000: Δισεκατομμύρια Η/Υ

Πληθώρα εφαρμογών

- Αυτοματοποίηση
- Απλοποίηση επίλυσης προβλήματος
- Αύξηση αποτελεσματικότητας
- Μείωση κόστους

Κοινωνία Πληροφορικής

- Διαδίκτυο (Internet)
- Παγκόσμιος Ιστός Πληροφοριών (World Wide Web)

Τεχνολογία Λογισμικού

Η συστηματική εφαρμογή διαδικασιών, μεθόδων, εργαλείων και τεχνικών για την επίτευξη μιας καθορισμένης απαίτησης ενός συστήματος λογισμικού

Ιστορικά στοιχεία

- 1968 NATO Conference
- Στόχος: η επίλυση της «κρίσης» του λογισμικού (software crisis)

Χτίσιμο μιας γέφυρας vs. «Χτίσιμο» ενός λειτουργικού συστήματος

- Τι γίνεται σε περίπτωση κατάρρευσης (collapse);
- Πολυπλοκότητα
- Συντήρηση

Κύκλος Ζωής Λογισμικού (Software Life Cycle)

Ο τρόπος που παράγουμε λογισμικό. Περιλαμβάνει:

- Μοντέλο κύκλου ζωής
- Άτομα που εμπλέκονται: αναλυτές, προγραμματιστές, managers, ...
- Εργαλεία ανάπτυξης λογισμικού (CASE *)

*CASE: Computer Aided Software Engineering

Κύκλος Ζωής Λογισμικού (Software Life Cycle)

Βασικές φάσεις κύκλου ζωής:

- **Ανάλυσης** Απαιτήσεων (Requirements)
- Καθορισμού Προδιαγραφών (Specifications)
- **Σχεδιασμού** (Design)
- **Υλοποίησης** (Implementation)
- Συνένωσης Κώδικα (Integration)
- Συντήρησης (Maintenance)
- Απόσυρσης (Retirement)

Μοντέλα Κύκλου Ζωής Λογισμικού

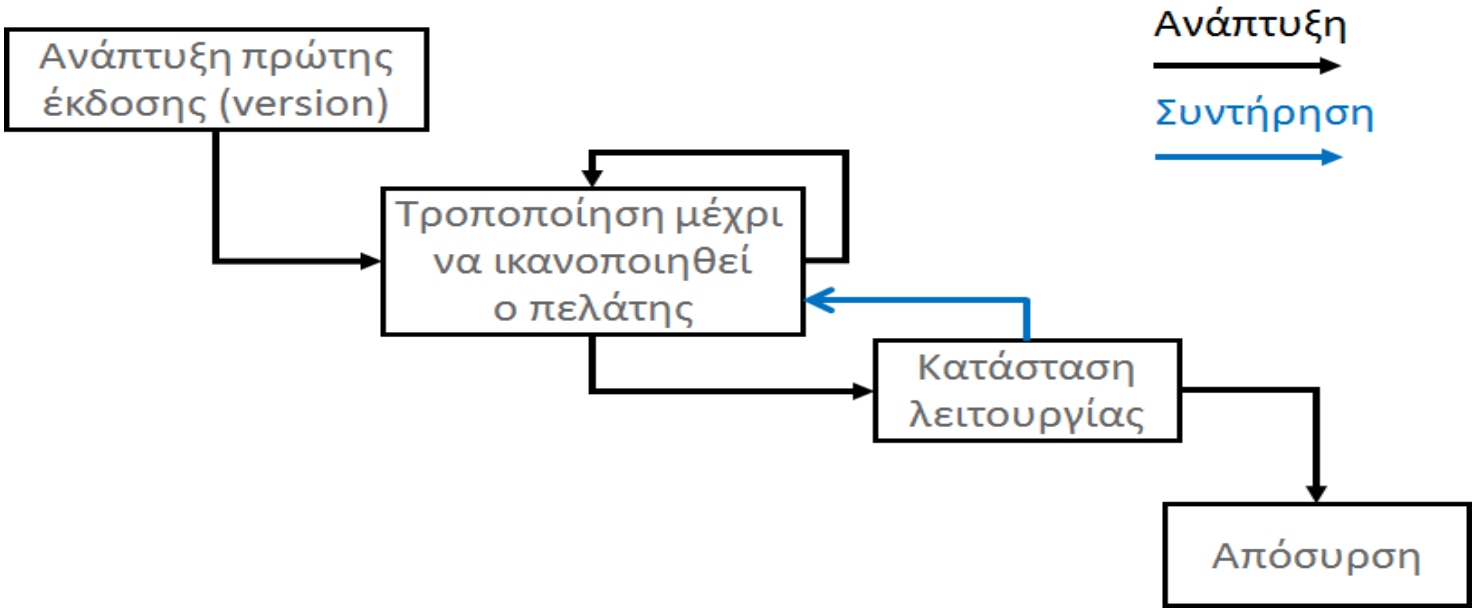
Αφορούν τη σειρά των βημάτων μέσω των οποίων ένα προϊόν λογισμικού αναπτύσσεται.

Παρέχουν οδηγίες για τις εργασίες που πρέπει να γίνουν, τη χρονική τους σειρά και τα κριτήρια μετάβασης από τη μία στην άλλη.

Υπάρχει ποικιλία μοντέλων:

- Όλα έχουν δυνατά σημεία και αδύνατα σημεία
- Κριτήρια επιλογής
  - οργάνωση
  - management
  - προσωπικό
  - προϊόν

Μοντέλο «build and fix»



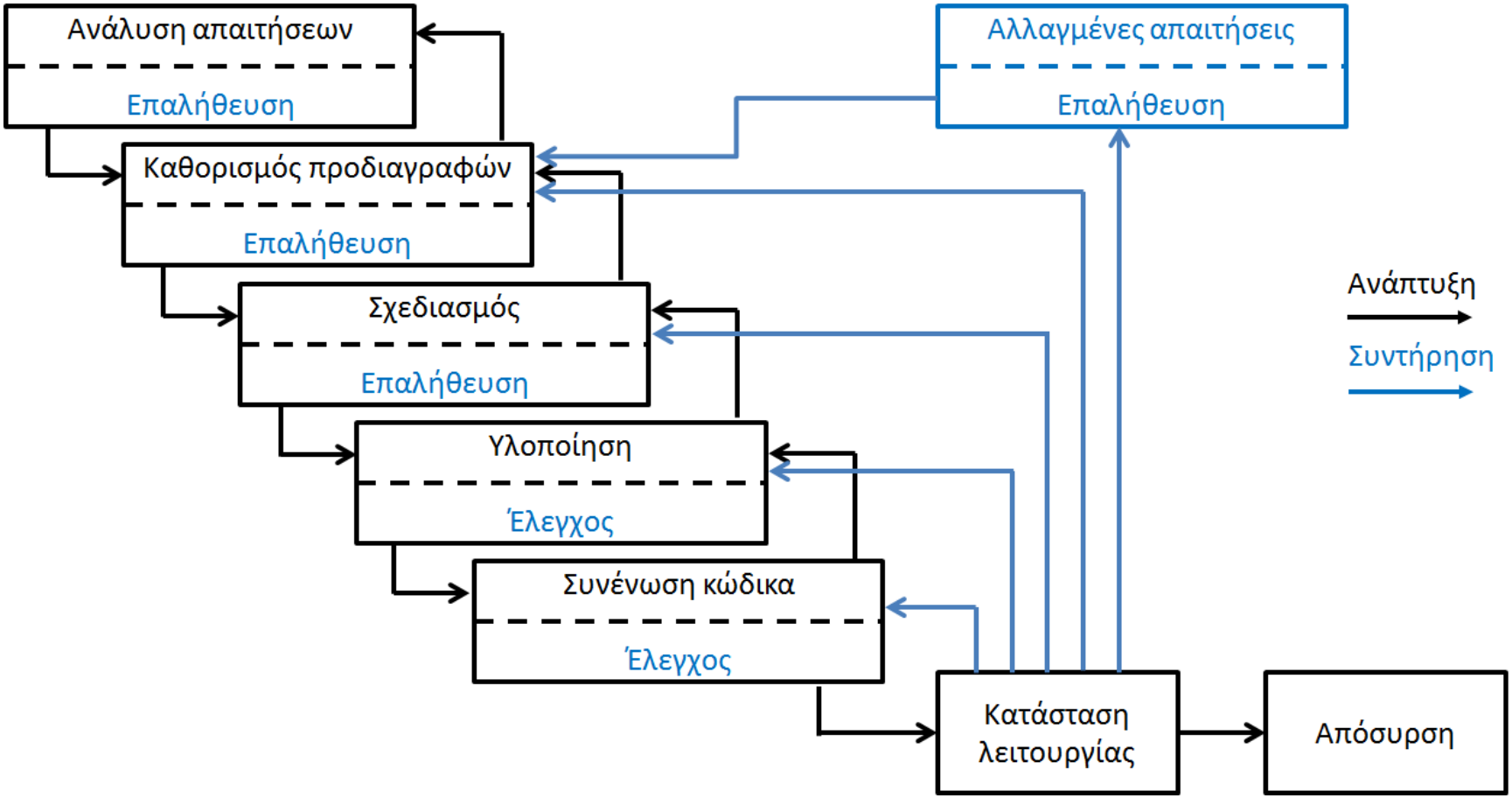
Εικόνα 1: Λογισμικό μοντέλο “build and fix”

Δεν γίνεται καθορισμός προδιαγραφών

Δεν γίνεται σχεδιασμός

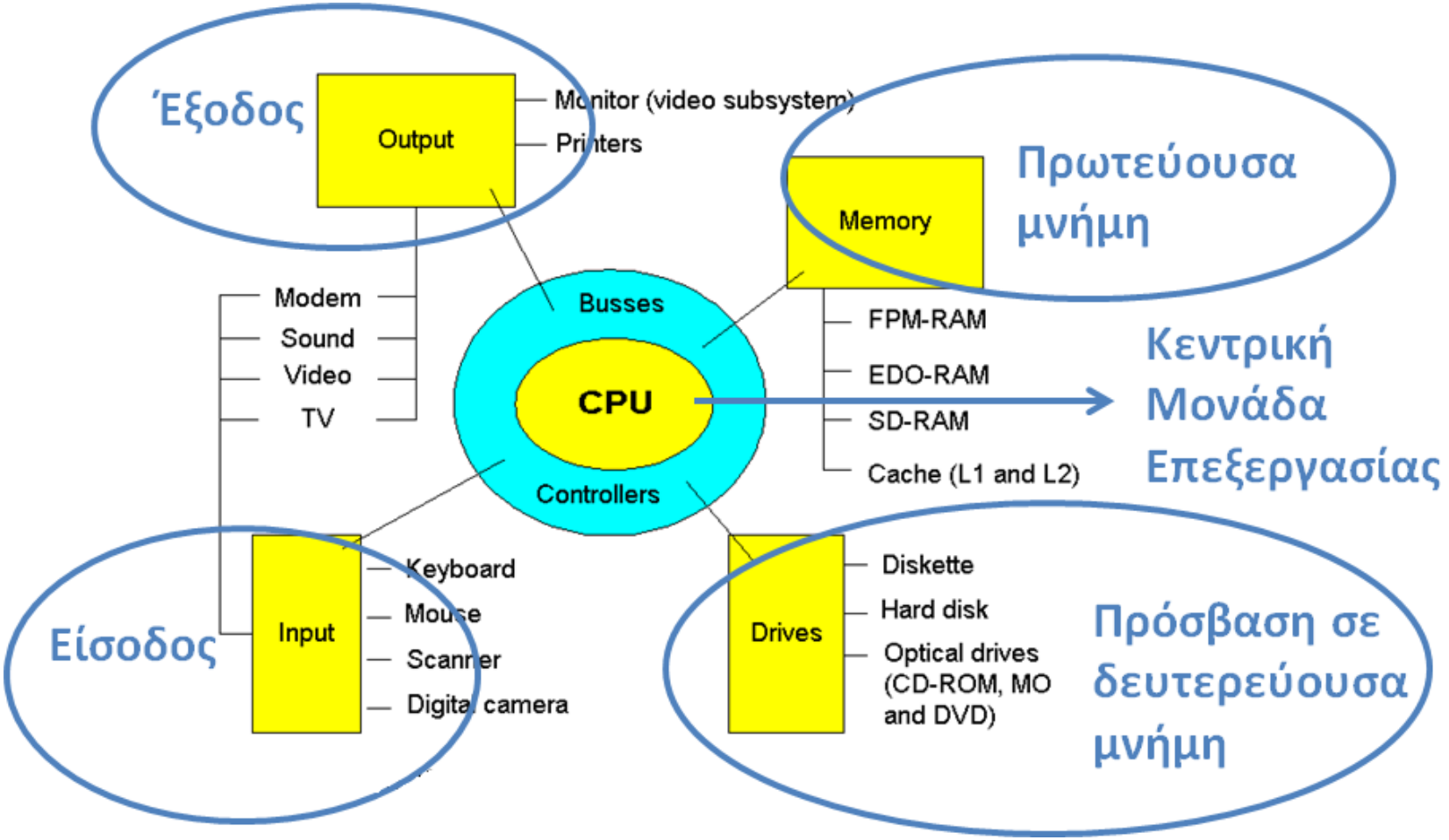
Δουλεύει καλά μόνο σε περιπτώσεις που το λογισμικό δεν ξεπερνά τις 100-200 γραμμές κώδικα

Μοντέλο καταρράκτη (waterfall model)



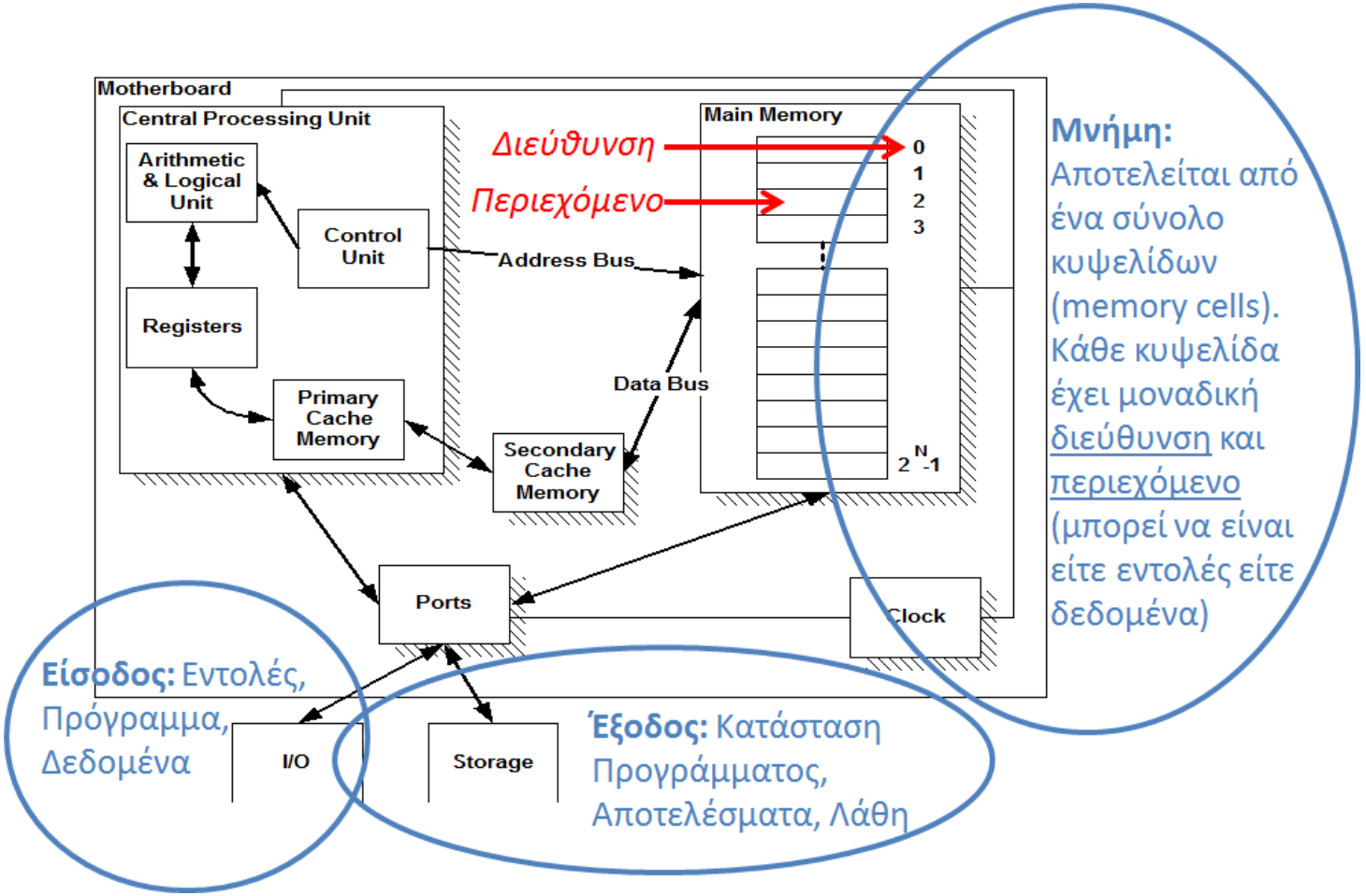
Εικόνα 2: Λογισμικό μοντέλο “καταρράκτη”

Αρχιτεκτονική ενός Η/Υ



Εικόνα 3: Σχέδιο λειτουργίας του Η/Υ

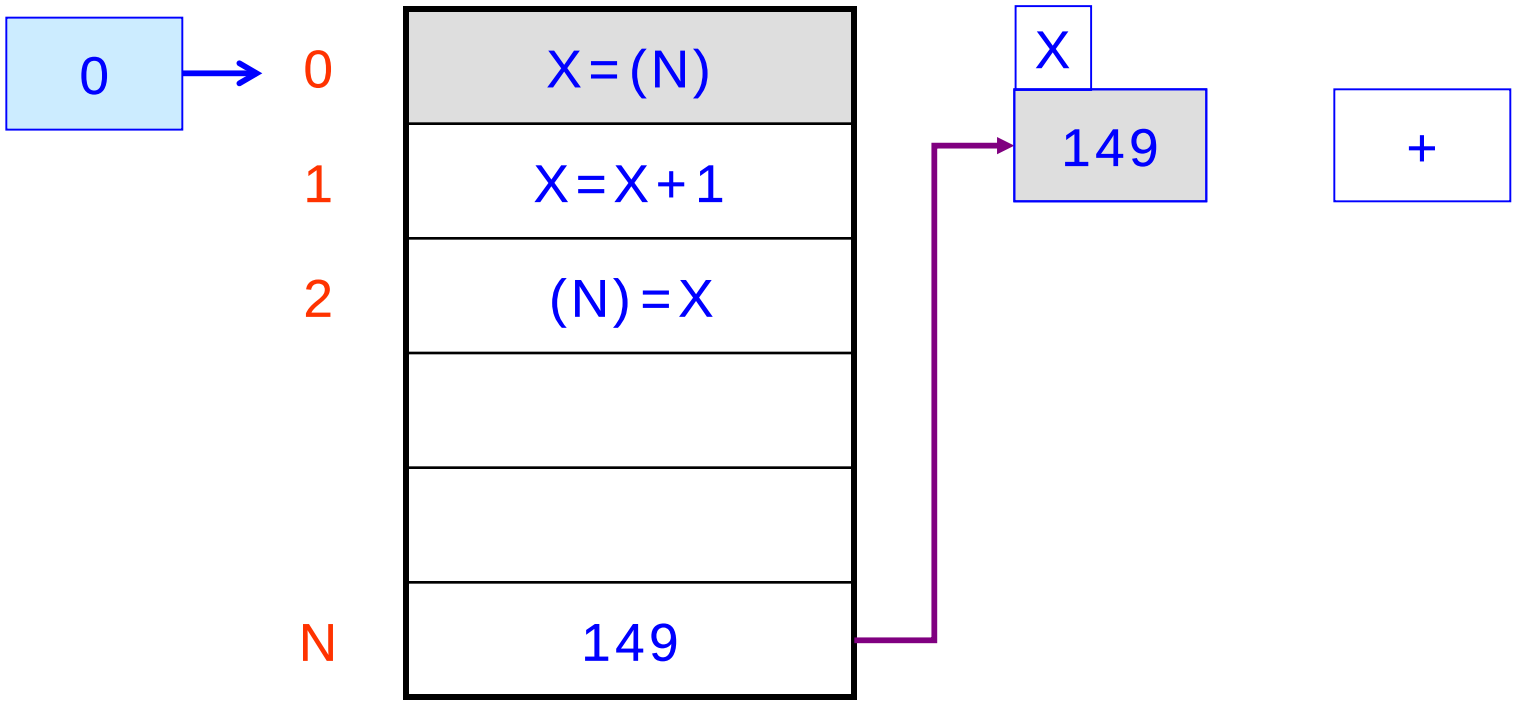
Μητρική πλακέτα (motherboard)



Εικόνα 4: Η Μητρική Πλακέτα του Η/Υ

Βασική λειτουργία ΚΜΕ

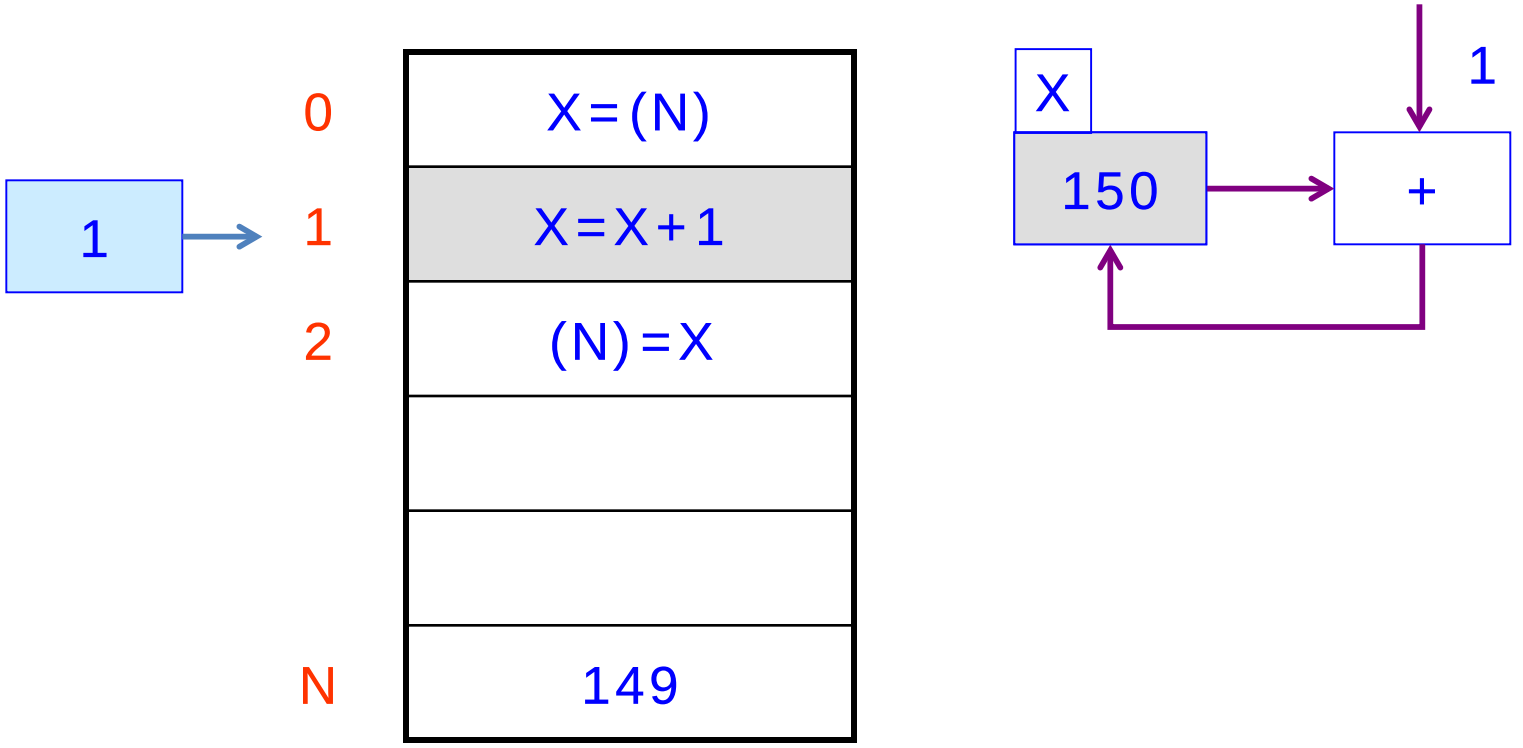
Κύκλος Προσκόμισης και Εκτέλεσης Εντολής (Fetch-Execute Cycle)



Εικόνα 5: Πρώτο βήμα εκτέλεσης μιας εντολής

Βασική λειτουργία ΚΜΕ

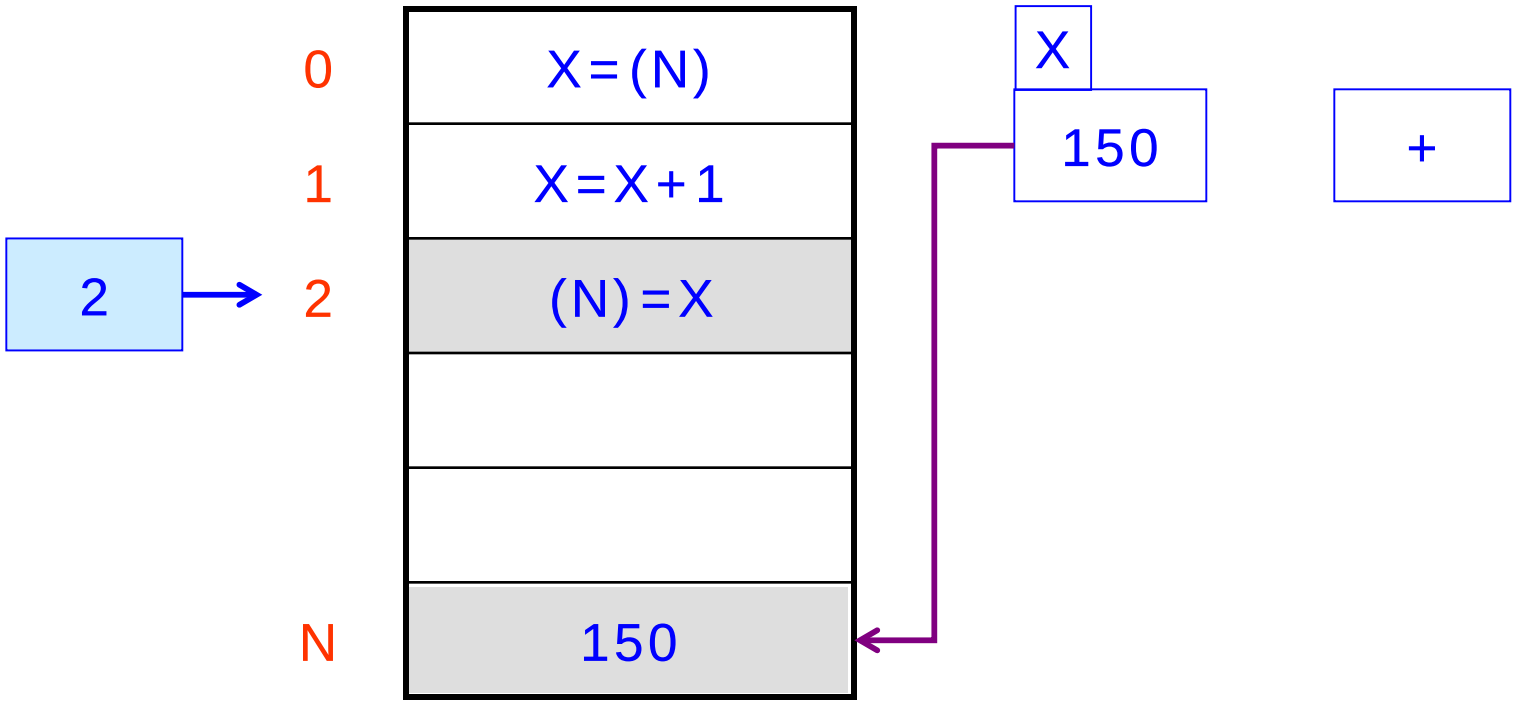
Κύκλος Προσκόμισης και Εκτέλεσης Εντολής (Fetch-Execute Cycle)



Εικόνα 6: Δεύτερο βήμα εκτέλεσης μιας εντολής

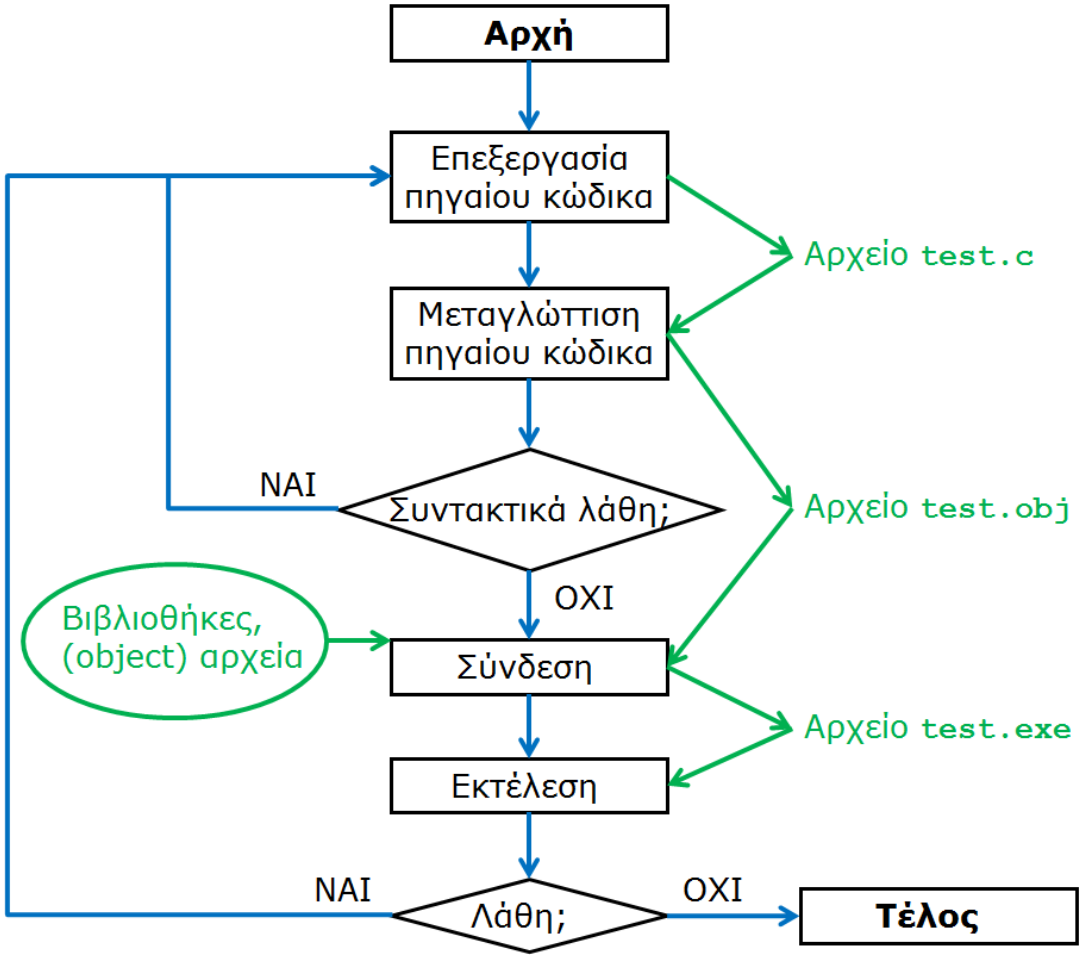
Βασική λειτουργία ΚΜΕ

Κύκλος Προσκόμισης και Εκτέλεσης Εντολής (Fetch-Execute Cycle)



Εικόνα 7: Τρίτο βήμα εκτέλεσης μιας εντολής

Φάση Υλοποίησης



Εικόνα 8: Διάγραμμα υλοποίησης προγράμματος

Μεταγλώττιση

Μεταγλωττιστής (compiler):

- Λογισμικό το οποίο μεταφράζει γλώσσα υψηλού επιπέδου (στην οποία είναι γραμμένη ένα πρόγραμμα) σε γλώσσα μηχανής (η οποία είναι «κατανοητή» από τον Η/Υ)
- Εξαρτάται από τη γλώσσα
- Παράμετροι (βλ. εγχειρίδιο χρήσης)

<όνομα μεταγλωττιστή> [<λίστα παραμέτρων>] <όνομα αρχείου>

```
> wcc test
> gcc -g test.c
```

Μεταγλώττιση

Πλήρως αυτοματοποιημένη διαδικασία

```
/* Ένα απλό πρόγραμμα σε C* /
#include <stdio.h>
main()
{
 printf(“hello world”);
}
```

Ανεύρεση λαθών:

- Παράλειψη τελευταίας αγκύλης

```
> Error! E1077: Missing ‘}’

> Parse error at end of input
```

Σύνδεση

Ακολουθεί τη μεταγλώττιση:

- Είναι κι αυτή πλήρως αυτοματοποιημένη
- Στα περισσότερα περιβάλλοντα ανάπτυξης, καλείται αυτόματα (χωρίς μεσολάβηση του προγραμματιστή) μετά από επιτυχημένη μεταγλώττιση
    - Περίπτωση gcc

Αποτέλεσμα σύνδεσης:

- Δημιουργία εκτελέσιμου (executable) κώδικα, ή
- Αναφορά προβλημάτων
    - Π.χ. αδυναμία εντοπισμού μιας συνάρτησης ή μιας εξωτερικής μεταβλητής

Σύνδεση

Δυνατότητες:

- Σύνδεση περισσοτέρων του ενός αρχείων αντικείμενου (object) κώδικα
- Αναζήτηση συναρτήσεων (π.χ. **printf**) σε βιβλιοθήκες
    - **#include <stdio.h>**

Μέρος 2ο

Στυλ και Γλώσσες Προγραμματισμού

Στυλ και Γλώσσες Προγραμματισμού

Βασικές Μορφές Προγραμματισμού

Βασικές μορφές (στυλ) προγραμματισμού

Διαδικασιακός ή Προστακτικός:

- Δεδομένα και ενέργειες (actions)
- C, Pascal, Ada, Algol, **Fortran**

Αντικειμενοστραφής:

- Αντικείμενο (object)
- Κληρονομικότητα, π.χ. C++, Smalltalk, Java, **Simula**

Βασικές μορφές (στυλ) προγραμματισμού

Συναρτησιακός:

- Συνάρτηση (function)
- **Lisp**, Miranda, ML

Λογικός:

- Μηχανισμός εξαγωγής συμπερασμάτων
- **Prolog**

Προσοχή:
Στην πράξη, μια γλώσσα προγραμματισμού σπάνια υποστηρίζει μια μόνο μορφή προγραμματισμού

Γλώσσες Προγραμματισμού

Γλώσσες Υψηλού Επιπέδου (High Level Language):

- Ανεξάρτητες του Η/Υ
- Συνδυάζουν αγγλικές λέξεις και «συνήθεις» μαθηματικούς συμβολισμούς
- Συχνά, είναι προσανατολισμένες στην επίλυση προβλημάτων συγκεκριμένου τύπου
    - COBOL: επεξεργασία επιχειρησιακών δεδομένων
    - BASIC: Beginners All-purpose Symbolic Instruction Code

Γλώσσες Προγραμματισμού

Γλώσσες Χαμηλού Επιπέδου (Low Level Language):

- Κατανοητές από (και «εξαρτημένες» με) συγκεκριμένη Κεντρική Μονάδα Επεξεργασίας
  - Γλώσσα Μηχανής (machine language) → Γλώσσα δυαδικών εντολών
  - Συμβολική Γλώσσα (assembly language) → Χρήση μνημονικών κωδικών που αντιστοιχούν σε εντολές της γλώσσας μηχανής (δύσχρηστη)

```
LOAD   MASS
MPY    ACCEL_CONST
STORE  FORCE
```

Γιατί C;

Πλεονεκτήματα:

- Γλώσσα προγραμματισμού χαμηλού επιπέδου (Άμεση πρόσβαση στους πόρους του Η/Υ)
- Γλώσσα προγραμματισμού υψηλού επιπέδου (Πληθώρα διαθέσιμων βιβλιοθηκών)
- Μικρή (σχετικά) και εύκολη στην εκμάθηση
- Υποστήριξη top-down και αρθρωτού (modular) σχεδιασμού
- Υποστήριξη δομημένου (structured) προγραμματισμού

Γιατί C;

Πλεονεκτήματα:

- Αποτελεσματική (Συμπαγή και γρήγορα στην εκτέλεση προγράμματα)
- Φορητή, ευέλικτη και με ισχυρές δυνατότητες
- Ευρύτατα χρησιμοποιούμενη γλώσσα (Μαζί με τις C++ και Java)
- Πολύ καλό εφόδιο για το «πέρασμα» στον αντικειμενοστραφή προγραμματισμό

Προγραμματισμός

Ανάπτυξη **αλγορίθμων** και υλοποίηση τους σε **γλώσσες προγραμματισμού** για την επίλυση **προβλημάτων**

Μεθοδολογία:

- Ορισμός προβλήματος
- Ανάλυση προβλήματος
  - Δεδομένα (inputs), αποτελέσματα, εξαγόμενα (outputs)
  - Τι πρέπει να γίνει

Προγραμματισμός

- Ανάπτυξη αλγορίθμου επίλυσης προβλήματος
- Υλοποίηση αλγορίθμου (πρόγραμμα)
- Δοκιμή και επαλήθευση
  - Αποσφαλμάτωση (προηγούμενα βήματα)

- Συντήρηση
  - Διορθωτική
  - Επαυξητική
    - Τελειοποίησης
    - Προσαρμογής

Αλγόριθμος

Πεπερασμένη σειρά ενεργειών, αυστηρά καθορισμένων και εκτελέσιμων σε πεπερασμένο χρόνο, που στοχεύουν στην επίλυση ενός προβλήματος

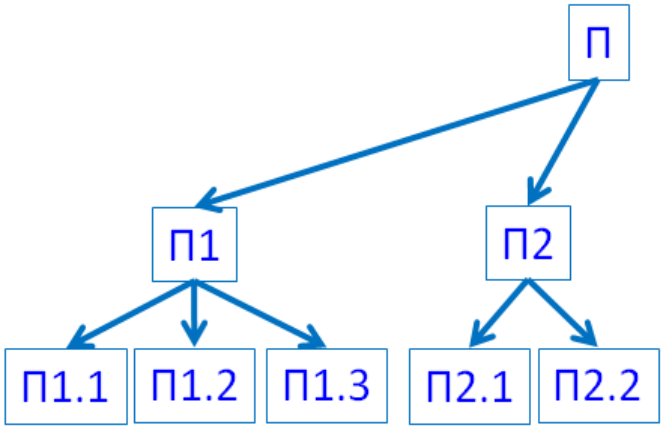
- Αναπαράσταση / διατύπωση της διαδικασίας **μετασχηματισμού** δεδομένων στα επιθυμητά αποτελέσματα

Ψευδοκώδικας (pseudocode)

Αλγόριθμος

Σχεδιασμός αλγορίθμου:

- Σταδιακή εκλέπτυνση (stepwise refinement) από πάνω προς τα κάτω (top-down design)
    - Το αρχικό πρόβλημα διασπάται σταδιακά σε απλούστερα προβλήματα
    - Η σταδιακή αυτή διάσπαση (εκλέπτυνση) συνεχίζεται μέχρις ότου φτάσουμε σε προβλήματα που δεν είναι δυνατό να διασπασθούν περαιτέρω
- Επαλήθευση και Αξιολόγηση



Εικόνα 9: Σχεδιασμός απλούστευσης του αρχικού προβλήματος

Σταδιακή εκλέπτυνση

Σπονδυλωτός σχεδιασμός (modular design)

Αφαιρετικότητα (abstraction):

- Διαχωρισμός ανάμεσα στο **τι** και το **πώς**
- Διάφορα επίπεδα αφαιρετικότητας

Διατύπωση διαπροσωπειών (interface) και διασυνδέσεων:

- Παράμετροι εισόδου και εξόδου (input/output, I/O)
- Λειτουργικότητα

Σταδιακή εκλέπτυνση

Παράλληλη ανάπτυξη (parallel development)

Επαναχρησιμοποίηση (reuse)

Επαλήθευση και Αξιολόγηση

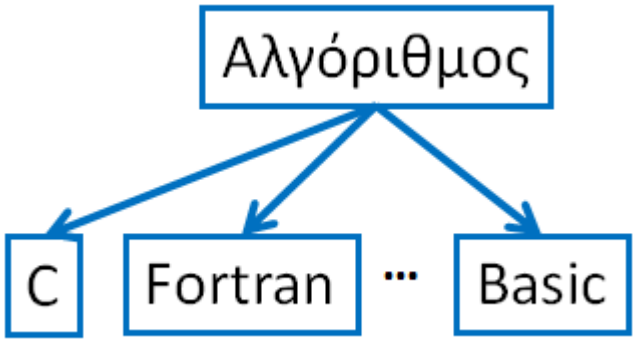
Αφαιρετικότητα

Αφαιρετικότητα μεθοδολογίας:

- Ορισμός προβλήματος
- Ανάλυση προβλήματος
- Ανάπτυξη αλγορίθμου επίλυσης προβλήματος

} ***Ανεξάρτητα της γλώσσας προγραμματισμού***

- Υλοποίηση αλγορίθμου (πρόγραμμα)
- Δοκιμή και επαλήθευση
- Συντήρηση



Εικόνα 10: Ανάπτυξη αλγορίθμου για την επίλυση του προβλήματος

Αφαιρετικότητα

Αφαιρετικότητα αλγορίθμου:

- Τι κάνει κι όχι πώς θα υλοποιηθεί

Αφαιρετικότητα προγράμματος:

- Τι κάνει ένα πρόγραμμα κι όχι πώς το κάνει (Διαδικασίες και Συναρτήσεις)

Αφαιρετικότητα δεδομένων:

- Τι είναι τα δεδομένα κάποιας δομής ή τύπου κι όχι πώς είναι οργανωμένα

Αφαιρετικότητα εντολών:

- Τι κάνει μια εντολή κι όχι πώς το κάνει

Ένα απλό παράδειγμα

Ορισμός Προβλήματος:

Να γραφεί πρόγραμμα που να υπολογίζει το εμβαδόν ενός ορθογωνίου:

Εμβαδόν = Μήκος x Πλάτος

Ανάλυση:

- Δεδομένα: δύο **πραγματικοί** αριθμοί
- Αποτέλεσμα: ένας **πραγματικός** αριθμός

Ένα απλό παράδειγμα

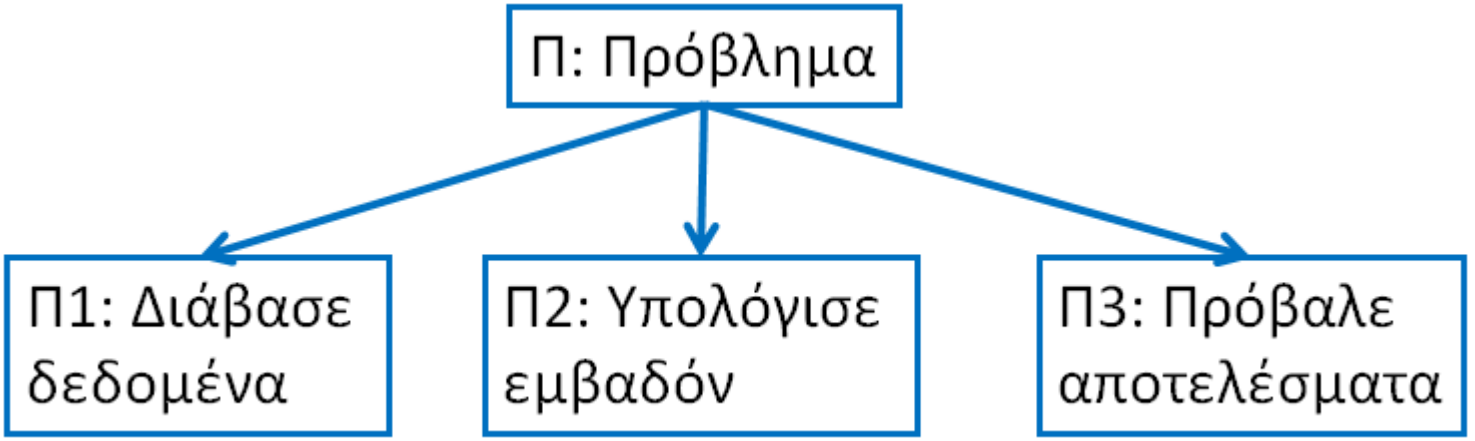
Ανάπτυξη αλγορίθμου:

- Διάβασε τα δεδομένα
- Υπολόγισε το εμβαδόν
- Προβολή αποτελέσματος

Ανάπτυξη αλγορίθμου:

- Διάβασε τα δεδομένα → διάβασε_δεδομένα(μήκος, πλάτος)
- Υπολόγισε το εμβαδόν → ε=υπολόγισε_εμβαδόν(μήκος, πλάτος)
- Προβολή αποτελέσματος → προβολή_αποτελ(μήκος, πλάτος, ε)

Ένα απλό παράδειγμα



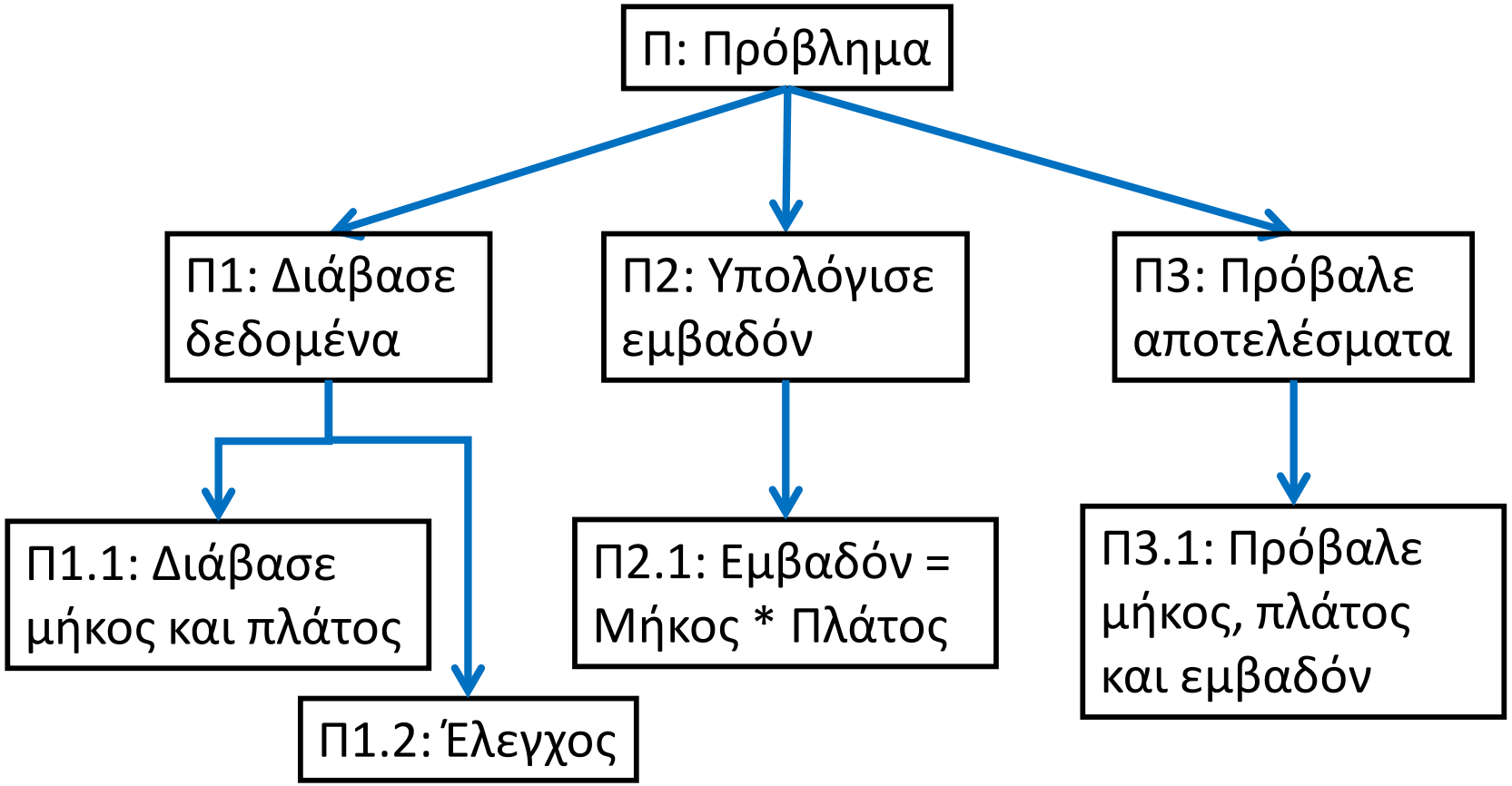
Εικόνα 11: Σχέδιο ανάπτυξης για την επίλυση του προβλήματος του παραδείγματος

Ένα απλό παράδειγμα

Πίνακας 1: Παράμετροι και Λειτουργικότητα

	Παράμετροι I/O	**Λειτουργικότητα**
Π1	Διαβάζει και επιστρέφει δύο τιμές (μήκος, πλάτος)	• Διάβασε μήκος και πλάτος • Έλεγξε αν είναι θετικά
Π2	• Λαμβάνει δύο παραμέτρους (μήκος, πλάτος) • Επιστρέφει εμβαδόν	Υπολόγισε εμβαδόν (εμβαδόν=μήκος*πλάτος)
Π3	Λαμβάνει τρεις παραμέτρους (μήκος, πλάτος, εμβαδόν)	Πρόβαλε το αποτέλεσμα

Ένα απλό παράδειγμα



Εικόνα 12: Ανάπτυξη του παραδείγματος

Ένα απλό παράδειγμα

Περαιτέρω εκλέπτυνση του Π1:

```
get(μήκος)          /* διάβασε μήκος */
get(πλάτος)         /* διάβασε πλάτος */

/* έλεγχος για μήκος */
if (μήκος<=0) print("λάθος τιμή για μήκος") exit

/* έλεγχος για πλάτος */
if (πλάτος<=0) print("λάθος τιμή για πλάτος") exit
```

Ένα απλό παράδειγμα

Υλοποίηση:

- Επιτυχής υλοποίηση προϋποθέτει γνώση γλώσσας προγραμματισμού
- Χρησιμοποίηση ονομάτων που βοηθούν στην εύκολη ερμηνεία του κώδικα

Τεκμηρίωση:

- Να συμπεριλαμβάνετε σχόλια στον κώδικα σας

Ένα απλό παράδειγμα

Εκτέλεση προγράμματος:

- Επαλήθευση και αξιολόγηση λύσεων (ταχύτητα κλπ.)
- Μηνύματα προς τον χρήστη
  - **Enter length in meters** 17
  - **Enter width in meters** 2
  - **The area of a rectangle with** 17 **m length and** 2 **m width is** 34**m²**
  - **Enter length in meters** -0.5
  - **Enter width in meters** 12
  - **error: length should be positive**

Τέλος Ενότητας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου των διδασκόντων.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Πατρών**» έχει χρηματοδοτήσει μόνο την αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημείωμα Αναφοράς

Νίκος Καρακαπιλίδης, Δημήτρης Σαραβάνος. «Προγραμματισμός Η/Υ. Εισαγωγή στον Προγραμματισμό». Έκδοση: 1.0. Πάτρα 2014. Διαθέσιμο από τη δικτυακή διεύθυνση: https://eclass.upatras.gr/courses/MECH1207/

Σημείωμα Αδειοδότησης

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.

Σημείωμα Χρήσης Έργων Τρίτων