ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν
Πανεπιστήμιον Αθηνών

Δομές Δεδομένων και Τεχνικές Προγραμματισμού

Ενότητα 1: Εισαγωγικές Έννοιες

Ιωάννης Κοτρώνης

Σχολή Θετικών Επιστημών

Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Σκοποί ενότητας

- Να εισάγει βασικές έννοιες, όπως αφαίρεση και απεικόνιση δεδομένων.
- Να εισάγει τον Αφαιρετικό Τύπο Δεδομένων (ΑΤΔ) και πώς υλοποιείται με ενότητες στην C.
- Να εισάγει στην αξιολόγηση αλγορίθμων με τον συμβολισμό O( ).

Περιεχόμενα ενότητας

- Εισαγωγή
- Επισκόπιση Μαθήματος
- Αφαιρετικοί Τύποι Δεδομένων (ΑΤΔ)
- Ενότητες στην C
- Αξιολόγηση Αλγορίθμων
- Συμβολισμός Ο( )

Ενότητα 1

Εισαγωγικές ‘Εννοιες

Αναζήτηση Αρχών στην Πληροφορική (Αφαίρεση- Απεικόνιση)

- Δύσκολη Πρόβλεψη για το μέλλον
- Όμως η Ανασκόπηση βοηθά

Π.χ. Hardware

Παρελθόν (ηλεκτρομηχανικά συστήματα, λυχνίες, transistors, chips)

Μέλλον ;; (οπτικοί, βιολογικοί,...)

Επιβίωση Αρχών (1)

- Παρόλες τις αλλαγές στο H/W
  - οι βασικές εντολές μηχανής
    - απόδοση τιμής, test, branch, …
  - τα βασικά δομικά στοιχεία δεδομένων
    - Bits, bytes, words

παρέμειναν τα ίδια.

- Οι νόμοι της αριθμητικής και λογικής παρέμειναν αναλλοίωτοι

Επιβίωση Αρχών (2)

- Ότι έχει αναπτυχθεί πάνω από αναλλοίωτα στοιχεία επιβιώνουν

  Γλώσσες Προγραμματισμού (Cobol, Fortran, Lisp, C, Pascal). Οι μεταγλωττιστές τους αποδείχτηκαν μεταφέρσιμοι. Οι βασικές αρχές οργάνωσης τους (αλγόριθμοι, δομές δεδομένων) παρέμειναν ίδιες.

- Επιβίωσαν επειδή είχαν σωστό επίπεδο **αφαίρεσης**

Η αρχή της Αφαίρεσης

- Κρατάμε μόνο τα χαρακτηριστικά από αντικείμενα που μας ενδιαφέρουν
- Αγνοούμε δευτερεύοντα χαρακτηριστικά ιδιαίτερα για το κάθε αντικείμενο.
- Π.χ. «Αυτοκίνητο»

  Πάρκινγκ – αριθμός κυκλοφορίας αυτοκινήτου

  Υπουργείο Συγκοινωνιών – Πολλά άλλα στοιχεία,

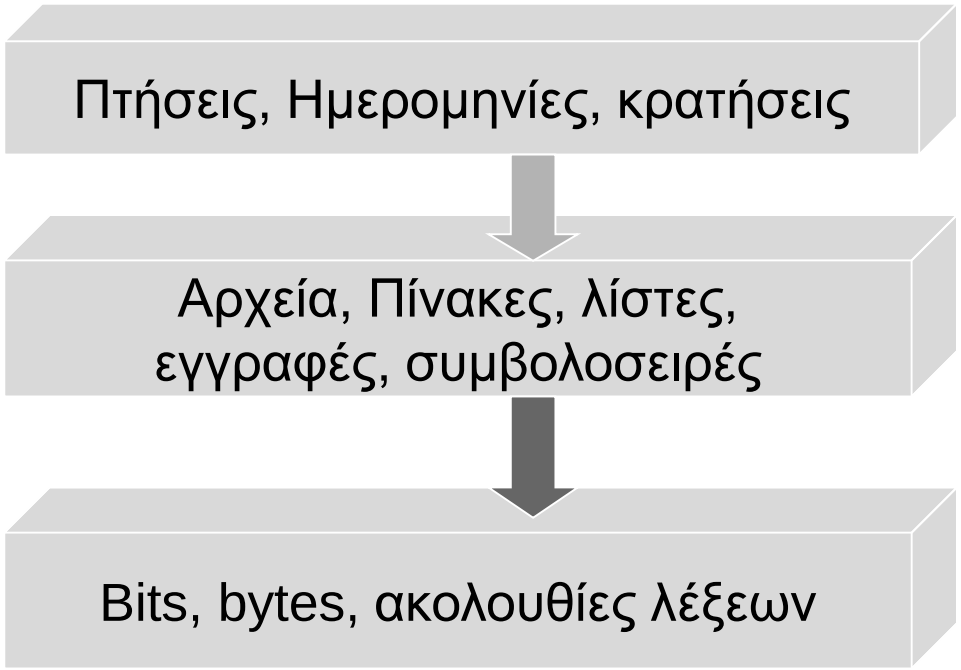
  Προσωπικά στοιχεία «πόσα καίει», service, ΚΤΕΟ κλπ

Οι σωστές Αφαιρέσεις επιβιώνουν

- Αφαιρέσεις που επέδειξαν Αξία και Χρησιμότητα στο χρόνο (παρόλες τις αλλαγές) τις δεχόμαστε ως θεμελιώδεις
- Πληροφορική (μια άποψη): Η αναζήτηση θεμελιακών αρχών υπολογισμού και η εφαρμογή τους.

Αρχές Δόμησης Λογισμικού

- Τι άλλες αρχές μπορούμε να «ανακαλύψουμε»;
- Π.χ. Αεροπορικές Πτήσεις



Η ανάπτυξη προγραμμάτων εμπεριέχει την **Απεικόνιση** Οντοτήτων, αντικειμένων, και συμπεριφοράς που απαιτούνται από την εφαρμογή χρησιμοποιώντας βασικά στοιχεία.

Ο Υπολογιστής είναι μια μηχανή απεικόνισης (Εικονική Πραγματικότητα)

Γιατί ενδιάμεσα επίπεδα; *Τελικά όλα καταλήγουν σε «bits» και «bytes»*

- Οι απαντήσεις είναι ακριβώς οι λόγοι ύπαρξης του μαθήματος και είναι πρωταρχικής σημασίας στην Πληροφορική.
- Στόχος η ανάπτυξη αξιόπιστων, αποδοτικών συστημάτων με ταχύτητα και οικονομία

Λόγοι για ενδιάμεσα επίπεδα

- **Οικονομία**. Κτίζουμε συστήματα από έτοιμα συστατικά (επαναχρησιμοποίηση)
- **Καλύτερη Διαχείριση**. Διασπούμε ένα μεγάλο πρόβλημα σε μικρότερα μέρη. Ιδανικά τα μέρη είναι υποσυστήματα που αλληλοεπιδρούν απλά, σύντομα και με διαυγή τρόπο.
- **Υψηλή χρηστικότητα**. Άθροιση εμπειρίας σε ρεπερτόρια ενδιάμεσων δομών.

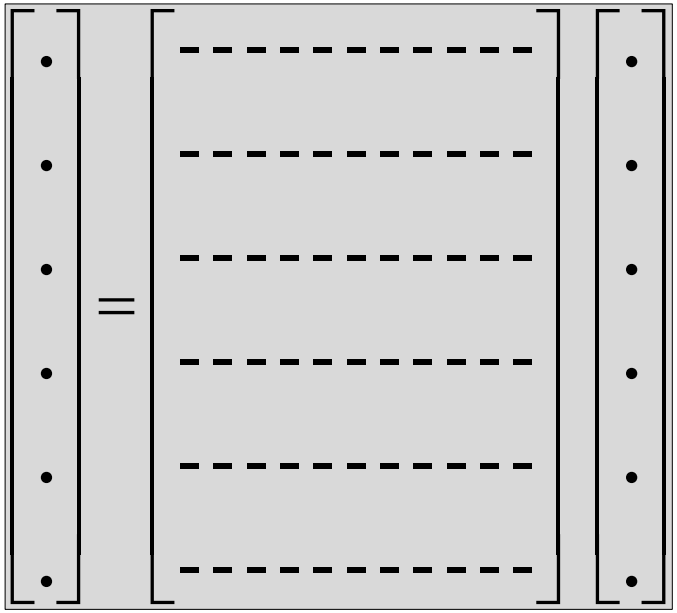
Αφαιρετικοί Τύποι Δεδομένων (ΑΤΔ)

- Στην Πληροφορική έχουμε πλήθος ενδιάμεσων Δομών Δεδομένων (συμβολοσειρές, πίνακες, εγγραφές, λίστες, δένδρα, ουρές, στοίβες, γράφους).
- Ο ΑΤΔ αναφέρεται στον τρόπο που «περιτυλίγουμε» (ορίζουμε και υλοποιούμε) τα Δεδομένα μαζί με τις επιτρεπτές πράξεις σε αυτά σε μια χρήσιμη ενότητα.
- Οι ΑΤΔ έχουν καθαρές διεπαφές («Τι κάνει») και χαμηλού επιπέδου απεικονίσεις («Πώς το κάνει»).
- Σας θυμίζει κάτι; (τύποι δεδομένων, συναρτήσεις)

Πολυπλοκότητα

- Οι πράξεις των ΑΤΔ υποστηρίζονται από αλγορίθμους.
- Η έννοια της πολυπλοκότητας των αλγορίθμων είναι σημαντική και θα μας απασχολήσει. «Πόσο καλός είναι;»
- Εκφράζουμε αυτή την πολυπλοκότητα με μαθηματικούς τύπους
- Κριτήρια μνήμης – ταχύτητας (συμβιβασμοί)

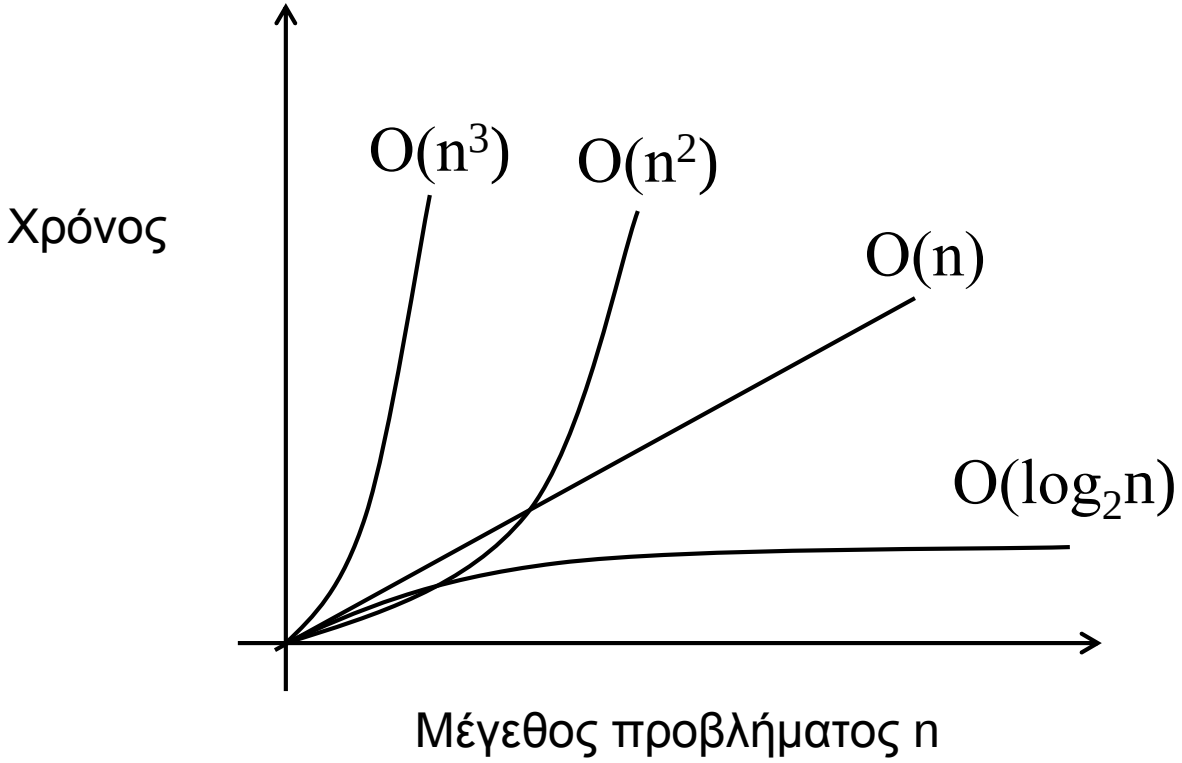
Y = A x



$$y = Ax, \quad y_i = \sum_{j=1}^{n} a_{ij}\, x_j, \quad i = 1(1)n$$

$$T(n) = [(n + (n-1)) * n] = 2n^2 - n = O(n^2)$$

Πολυπλοκότητα Αλγορίθμων ο συμβολισμός O( )



Κύριο Αντικείμενο Μαθήματος ΑΤΔ και Υλοποιήσεις στην C

Boolean

Στοίβα (Stack)

Ουρά (Queue)

Λίστες (Lists)

Δένδρα (Trees)

Γραφήματα (Graphs)

Εξίσου Σημαντικοί Άξονες (1)
Εργαστήρια

Προγραμματισμός

Εμπέδωση, εμβάθυνση στην C

Π.χ. Δυναμική Διαχείριση Μνήμης-δείκτες, Αναδρομή

Τεχνικές Προγραμματισμού

Ενότητες (modules .h + .c)

Δοκιμή και Εκσφαλμάτωση

Εξίσου Σημαντικοί Άξονες (2)

- Εισαγωγή στην Αξιολόγηση Αλγορίθμων
  - Πολυπλοκότητα Χρόνου-Μνήμης
- Γενικές Αρχές Πληροφορικής
  - Απεικόνιση-Αφαίρεση
  - Διεπαφές-Απόκρυψη Πληροφορίας
  - Επαναχρησιμοποίηση Κώδικα
  - κλπ

Ενότητες στην C

Αφαιρετικοί Τύποι Δεδομένων και Υλοποποίηση τους στην C

ΑΦΑΙΡΕΤΙΚΟΣ (ή ΑΦΗΡΗΜΕΝΟΣ) ΤΥΠΟΣ ΔΕΔΟΜΕΝΩΝ (ΑΤΔ)

(Abstract Data Type-ADT)

- Τύπος Δεδομένων:

  - σύνολο δεδομένων (data, objects)

  - σύνολο πράξεων στα δεδομένα

- Ένας ΑΤΔ είναι ένα μαθηματικό μοντέλο (οντότητα) που ορίζει ένα τύπο δεδομένων.

-Η έννοια του ΑΤΔ είναι θεωρητική (αφαιρετική) και έχει σαν σκοπό τον ορισμό

-των δεδομένων και

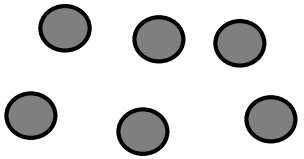
-των μεταξύ αυτών πράξεων

αγνοώντας τις λεπτομέρειες υλοποίησης του.

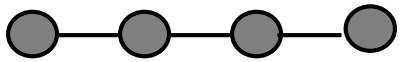
δομική σχέση στοιχείων ΑΤΔ

Βασικός τύπος
(boolean, int, float, …)

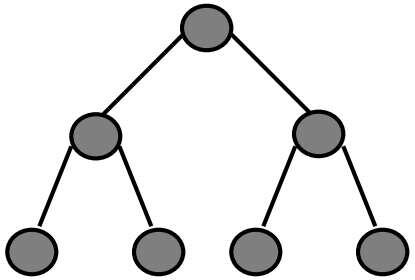
σύνολο (set)



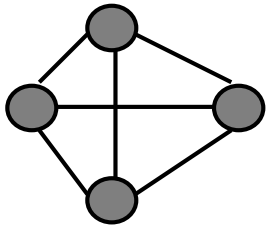
γραμμική (linear)
(στοίβα, ουρά, λίστα)



ιεραρχική (hierarchical)
(δένδρο, δυαδικό δένδρο)



δίκτυο (network)
(γράφημα)



Υλοποίηση του ΑΤΔ

• Απεικόνιση σε κάποια γλώσσα προγραμματισμού του τύπου που ορίζει μια μεταβλητή του ΑΤΔ (typedef … <new name>)

• υποπρογράμματα (συναρτήσεις) για κάθε πράξη του

Πλεονεκτήματα του ΑΤΔ

• Ορισμός της επεξεργασία δεδομένων σε ένα αφηρημένο επίπεδο (τι κάνει)

• Απόκρυψη πληροφορίας υλοποίησης (πώς το κάνει)

• Μελλοντικές αλλαγές στην υλοποίηση μπορούν να γίνουν ανεξάρτητα από το υπόλοιπο πρόγραμμα που χρησιμοποιεί τον ΑΤΔ

Παράδειγμα: Ο ΑΤΔ Λογικός (Boolean)

Δύο είναι οι τιμές του ΑΤΔ Λογικός οι :
true και false.

Οι βασικές του πράξεις είναι :

Καταχώρηση	Assign
Και	AND
Η (διεζευκτικό)	OR
Μη	NOT

Πιθανά και άλλες (π.χ. XOR, NAND)

Ο τύπος δεδομένων

typedef enum _BOOLEAN_ {F=0, T} BOOLEAN;

Οι πράξεις (επικεφαλίδες)

```
BOOLEAN OR      (BOOLEAN b1, BOOLEAN b2);
BOOLEAN AND     (BOOLEAN b1, BOOLEAN b2);
BOOLEAN NOT     (BOOLEAN b);
```

Και οι υλοποιήσεις τους ...

```
BOOLEAN OR (BOOLEAN b1, BOOLEAN b2)
{   if ((b1==T) || (b2==T))
            return T;
    else  return F;
}

BOOLEAN AND (BOOLEAN b1, BOOLEAN b2)
{   if ((b1==T) && (b2==T))
            return T;
    else  return F;
}

BOOLEAN NOT (BOOLEAN b)
{   if ((b==T))
            return F;
    else return T;
```

Ενότητες στην C

Τεχνική Υλοποίησης
Αφαιρετικών Τύπων Δεδομένων στην C

Δυσκολία:

Προγράμματα που λύνουν «πραγματικά προβλήματα» μπορεί να είναι μεγάλα (χιλιάδες ή εκατομμύρια γραμμές κώδικα). Κανείς δεν καταλαβαίνει ή θυμάται τόσο μεγάλα προγράμματα.

Πρόβλημα:

Πώς ξεπερνάμε τις δυσκολίες;

Δύο βασικές ιδέες

1. **Διαχώρισε το πρόβλημα σε μικρότερα καλώς καθορισμένα υπο-προβλήματα**

2. **Απόκρυψε πληροφορίες υλοποίησης, όπου είναι δυνατόν**

Τι σημαίνουν;

Διαχωρισμός (ΤΙ κάνει το κάθε τμήμα)

Η πρώτη ιδέα απαιτεί να μπορούμε να κολλήσουμε τα κομμάτια στα οποία έχουμε χωρίσει το μεγάλο πρόβλημα. Τα μικρότερα κομμάτια πρέπει να «μπουν μαζί» για να κατασκευάσουμε μεγαλύτερα κομμάτια ή και το τελικό πρόγραμμα.

Απόκρυψη υλοποίησης (ΠΩΣ το κάνει)

Η δεύτερη ιδέα μας επιτρέπει να ανακάμψουμε από μια ενδεχομένως κακή απόφαση υλοποίησης χωρίς να χρειάζεται να ξεκινήσουμε από την αρχή. Από την άλλη πλευρά μια καλή υλοποίηση μπορεί να ξανα-χρησιμοποιηθεί σε άλλα προγράμματα μειώνοντας τον χρόνο και το κόστος.

Ο μηχανισμός της C για να υλοποιήσουμε τις δύο πολιτικές είναι η

Ενότητα (**MODULE**)

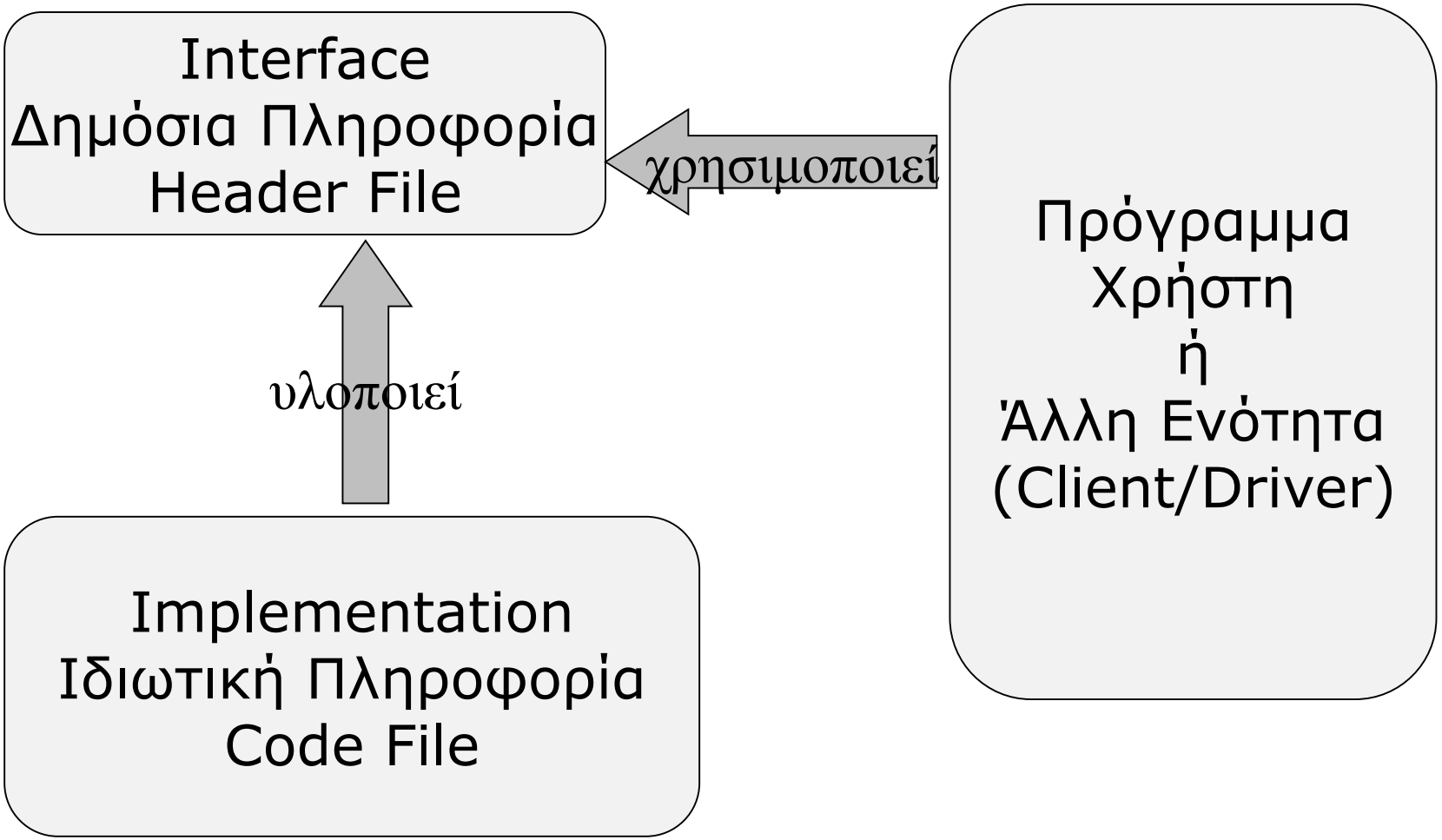
Σε φυσικό επίπεδο αποτελείται από δύο αρχεία: Αρχείο **Διεπαφής** (**InterfaceFile .h**)

Αρχείο **Υλοποίησης** (**ImplementationFile .c**)

Το αρχείο Διεπαφής διαθέτει την **Δημόσια** (**PUBLIC)** πληροφορία, δηλαδή την πληροφορία που χρειάζεται ο Χρήστης για να **χρησιμοποιήσει** την λειτουργικότητα.

Το αρχείο Υλοποίησης περιλαμβάνει την ιδιωτική πληροφορία, δηλαδή την υλοποίηση της λειτουργικότητας την οποία «δημοσιεύει» το αρχείο Διεπαφής.

Οργάνωση Προγράμματος σε Ενότητες (Πολλαπλά αρχεία)



Τι Είναι μια **Ενότητα** (**Module**)

Ένα σύνολο δηλώσεων που μπορούν να χρησιμοποιηθούν σε ένα πρόγραμμα.

Είναι μια μονάδα οργάνωσης ενός λογισμικού συστήματατος που

Α) **Έχει αυτοτέλεια**. Τοποθετούνται μαζί μια συλλογή από οντότητες (δεδομένα και πράξεις-συναρτήσεις) που ορίζουν ένα σύνολο δυνατοτήτων χρήσιμο στο να λύνει κάποια προβλήματα. (encapsulation – ενθυλάκωση)

Β) **Διαχωρίζει το Τι από το Πώς**. Προσδιορίζει τι δεδομένα ή πράξεις επιτρέπεται να βλέπουν και να χρησιμοποιούν οι εξωτερικοί χρήστες.

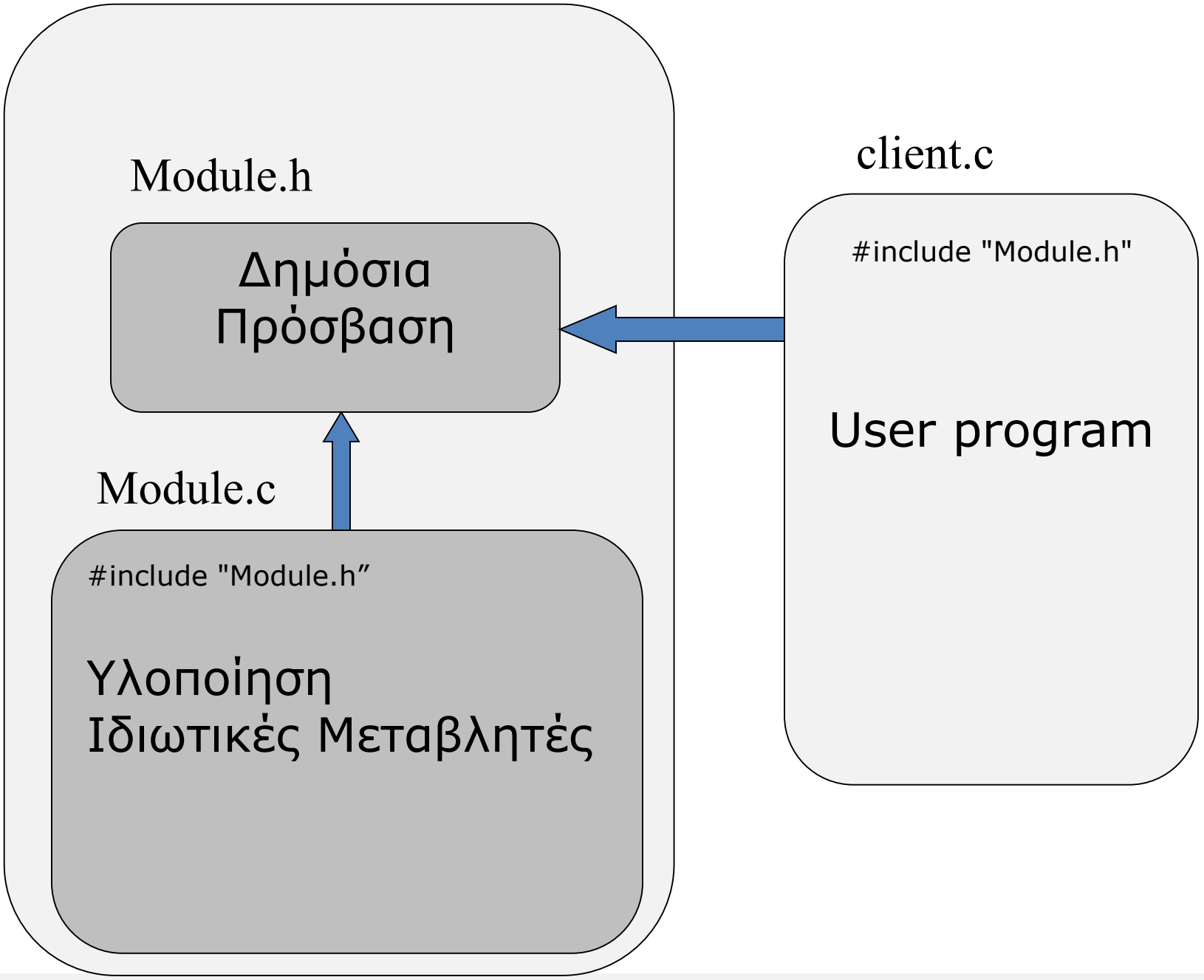
Πώς δημιουργούμε μια ενότητα

Διεπαφή **Module.h**

Ένα αρχείο που περιλαμβάνει όλες τις οντότητες που πρέπει να είναι ορατές: constants, type definitions, variable definitions, and functions (i.e., function prototypes) που ο χρήστης επιτρέπεται να χρησιμοποιήσει (συνάρτηση) ή αλλάξει (μεταβλητή).

Υλοποίηση **Module.c**

Ένα αρχείο που περιλαμβάνει όλες τις ιδιωτικές οντότητες: τον κώδικα υλοποίησης των συναρτήσεων και όλες τις σταθερές, μεταβλητές και συναρτήσεις που ο χρήστης της ενότητας δεν επιτρέπεται να έχει **άμεση** πρόσβαση.


Module.h
Δημόσια
Πρόσβαση
client.c
#include "Module.h"
User program
Module.c
#include "Module.h”
Υλοποίηση
Ιδιωτικές Μεταβλητές

Παράδειγμα Boolean
Interface file: Boolean.h

```
#ifndef __CH2_BOOLEAN__
#define __CH2_BOOLEAN__

    typedef enum BOOLEAN {F=0, T} BOOLEAN;

    void kataxorisi (BOOLEAN * const bPtr, BOOLEAN bexpr);

     BOOLEAN OR (BOOLEAN b1, BOOLEAN b2);
     BOOLEAN AND (BOOLEAN b1, BOOLEAN b2);
     BOOLEAN NOT (BOOLEAN b);

     int diabasma (BOOLEAN *bPtr);
     void grapsimo (BOOLEAN b);

#endif
```

Implementation file: Boolean.c

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

void kataxorisi (BOOLEAN * const bPtr, BOOLEAN bexpr)
{   *bPtr=bexpr;
}

BOOLEAN OR (BOOLEAN b1, BOOLEAN b2)
{   if ((b1==T) || (b2==T))
        return T;
    else return F;
}

BOOLEAN AND (BOOLEAN b1, BOOLEAN b2)
{   if ((b1==T) && (b2==T))
        return T;
    else return F;
}

BOOLEAN NOT (BOOLEAN b)
{   if ((b==T))
        return F;
    else return T;
}
```

Πώς χρησιμοποιούμε μια ενότητα (Module):

Το πρόγραμμα πελάτης (client) του Χρήστη κάνει δήλωση χρήσης μέσω include directive.

```
#include <stdio.h>          /* include system file */
/*  …. Other system inclusions   …. */
#include “Module.h” /* include non-system module */
/*  …. Other modules …. */
/*  ….  User Program …. */
```

Παράδειγμα Χρήσης Boolean

```
#include <stdio.h>
#include <stdlib.h>
#include "Boolean.h"

int main( )
{ BOOLEAN a=T, b=F, c; // μεταβλητές και αρχικοποίηση

  kataxorish(&c, OR(a,b));
  kataxorish(&c, AND(a,b);
  a=F;
  kataxorisi(&a, T);
  kataxorisi(&c, AND ( OR(a, NOT(b)), OR (NOT(a), b));
}
```

Πλεονεκτήματα

- Ξεχωριστή μεταγλώττιση Ενότητας και Προγράμματος Χρήσης (όχι άμεσα ορατό με την χρήση Dev C++ ή VisualStudio)

```
gcc.exe -c Boolean.c -o ch2_Boolean.o
gcc.exe -c main.c -o main.o
gcc.exe Boolean.o main.o  -o "Project1.exe"
```

Κοινό χαρακτηριστικό των ενοτήτων σε όλες τις γλώσσες που υποστηρίζουν αυτό τον μηχανισμό είναι η ξεχωριστή μεταγλώττιση (***Separate Compilation***)

Απλά σημαίνει ότι η συλλογή δεδομένων και συναρτήσεων μπορεί να μεταγλωττιστεί αυτόνομα από άλλες και από άλλα προγράμματα που την χρησιμοποιούν.

Αλλαγές στο πρόγραμμα του χρήστη ή σε μια από τις συλλογές απαιτεί την μεταγλώττιση ενός μικρού αριθμού ενοτήτων. Αυτό μπορεί να μην είναι σημαντικό για προγράμματα των Χ100 ή Χ1000 γραμμών, αλλά είναι κρίσιμο για προγράμματα Χ1.000.000

Αφαίρεση (Abstraction):

Αφαίρεση Διαδικασιών (Procedural Abstraction) – Πώς να αντικαταστήσουμε μια (μεγάλη) ακολουθία εντολών (πώς το κάνει) με ένα Όνομα και μια διεπαφή (τί κάνει). Ενσωματώνεται στις Συναρτήσεις (FUNCTION) με καλά καθορισμένες παραμέτρους και τύπο επιστροφής (return)

Απόκρυψη Πληροφορίας (τοπικές μεταβλητές)

Αλλαγή Υλοποίησης (Αλλαγές μόνο στο boolean.c)

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

void kataxorisi (BOOLEAN * const bPtr, BOOLEAN bexpr)
{   *bPtr=bexpr;
}

BOOLEAN OR (BOOLEAN b1, BOOLEAN b2)
{   return (b1||b2); /* πριν if ((b1==T) || (b2==T)) return T; else return F; */
}

BOOLEAN AND   (BOOLEAN b1, BOOLEAN b2)
{  return (b1&&b2);
}

BOOLEAN NOT   (BOOLEAN b)
{ return (!b);
}
```

Τέλος Ενότητας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στο πλαίσιο του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Αθηνών**» 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.

Σημειώματα

Σημείωμα Αναφοράς

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.