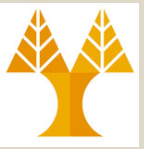
Τμήμα Πληροφορικής
Πανεπιστήμιο Κύπρου



ΕΠΛ232 – Προγραμματιστικές Τεχνικές και Εργαλεία

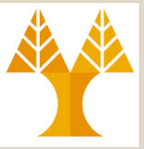
Διάλεξη 1: Εισαγωγή - C για Προγραμματιστές JAVA

(Κεφάλαια 1-2, KNK-2ED)

Δημήτρης Ζεϊναλιπούρ

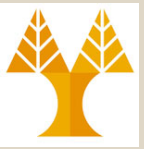
http://www.cs.ucy.ac.cy/courses/EPL232

Στόχοι Μαθήματος



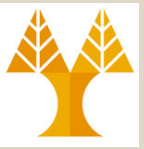
- **ΕΠΛ131:** Αρχές Προγραμματισμού Ι
  - Ανάπτυξη δεξιοτήτων στην **επίλυση προβλημάτων** με αλγοριθμικό τρόπο, μέσω **διαδικαστικού** και **αντικειμενοστρεφούς προγραμματισμού**
  - Θεμελίωση της **αλγοριθμικής σκέψης** και των **βασικών αρχών προγραμματισμού**
- **ΕΠΛ232:** Προγραμματιστικές Τεχνικές και Εργαλεία
  - Ενδιάμεσες και **προχωρημένες έννοιες και τεχνικές προγραμματισμού** μέσω μιας χαμηλού επιπέδου γλώσσας
  - Ανάπτυξη μεγάλων **εύρωστων προγραμμάτων / βιβλιοθηκών** τα οποία θα επιλύνουν **πολύπλοκα προβλήματα.**
  - Προχωρημένα θέματα **διαχείρισης της κύριας** και **δευτερεύουσας** μνήμης από τη γλώσσα προγραμματισμού, θέματα **μεταγλώττισης**, **ολοκληρωμένα εργαλεία ανάπτυξης**, μεθόδους **αποσφαλμάτωσης** και **βελτιστοποίησης** του κώδικα

Συμβόλαιο Μαθήματος



- **Επίπεδο:** Προπτυχιακό
  - Υποχρεωτικό Μάθημα
- **Πίστωση:** 7.5 μονάδες ECTS
- **Προαπαιτούμενα:**
  - ΕΠΛ131: Αρχές Προγραμματισμού Ι
- **Μέθοδοι Διδασκαλίας**
  - **Διαλέξεις & Φροντιστήρια** (5 ώρες εβδομαδιαίως): Συνδυασμένη Παράδοση Διδακτέας Ύλης και Θεωρητική Εμπέδωση
  - **Εργαστήριο** (2 ώρες εβδομαδιαίως): **Εμπέδωση διαλέξεων μέσω ασκήσεων,** εκμάθηση εργαλείων, πρακτική εξάσκηση, ευκαιρία για πιο προσωπική επίλυση απορίων.

Συμβόλαιο Μαθήματος



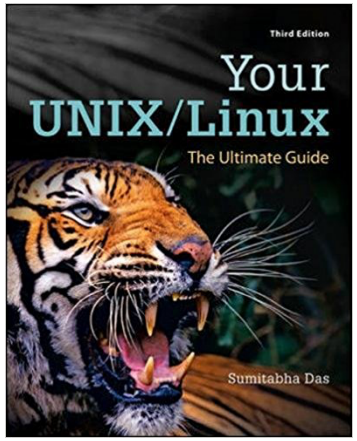
- **Αξιολόγηση**
  - 55% Τελική Εξέταση (1)
  - 20% Ενδιάμεση Εξέταση (1)
    - **Ημερ.: Τρίτη, 23 Οκτωβ. 2018! (8η Εβδ.)**
  - 25% Ασκήσεις
    - Προγραμματιστικές Ασκήσεις (4)
    - Ομαδική Εργασία (1)

Βιβλιογραφία

Βασική Βιβλιογραφία

- K.N. King, *C Programming: A Modern Approach*, Second Edition, ISBN-10: 0393979504, ISBN-13: 978-0393979503, 832 , W. W. Norton & Company, 2008.
- Your UNIX/Linux: The Ultimate Guide, 3rd Edition, Sumitabha Das, McGraw Hill, ISBN-13 9780073376202, 800 pages, 2013.

Βοηθητική Βιβλιογραφία

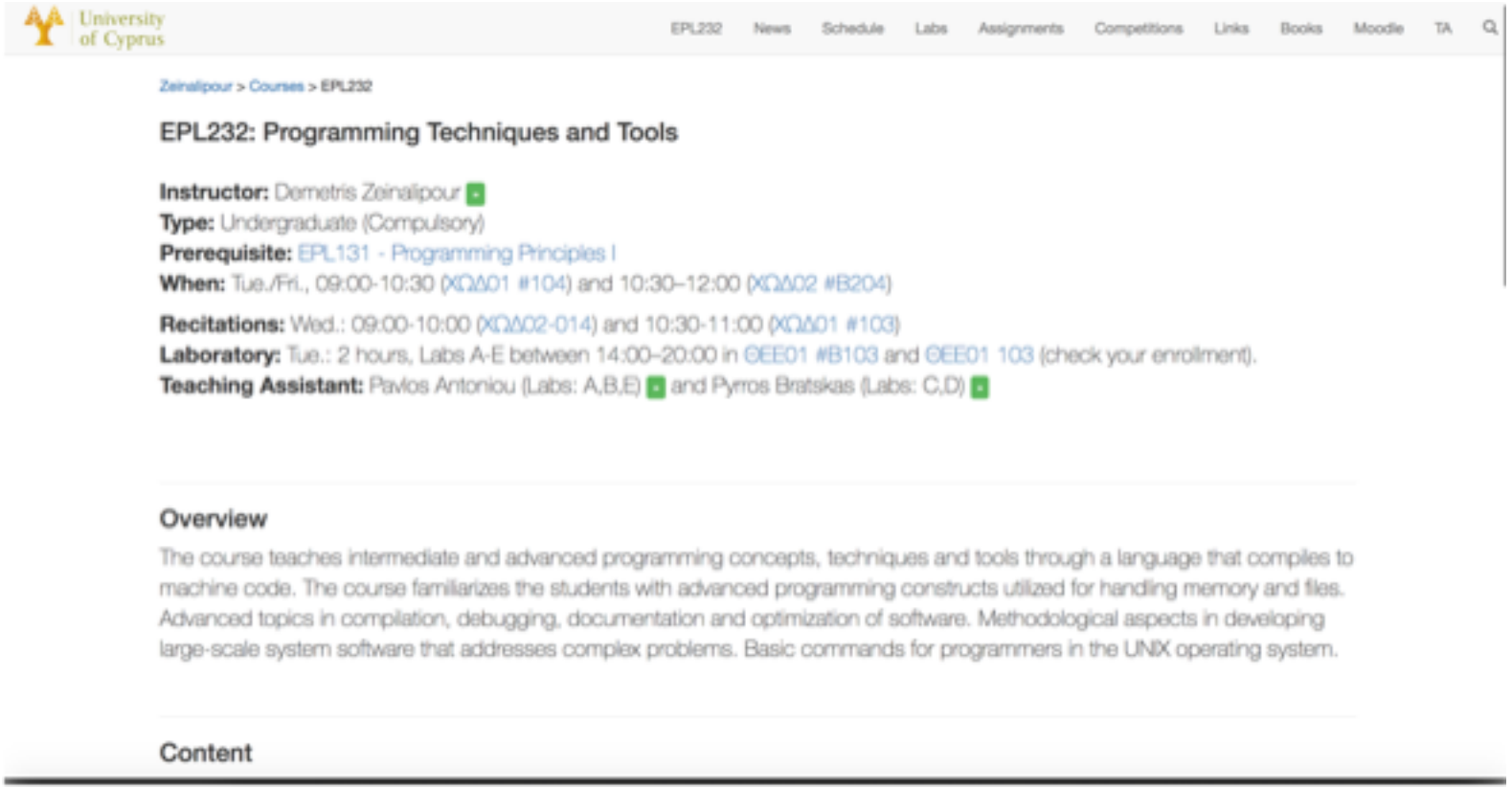


- Σημειώσεις Διαλέξεων Μαθήματος
- Programming in C, 4th Edition, Stephen G. Kochan, ISBN-10: 0321776410, ISBN-13: 9780321776419, Addison-Wesley Professional, 600 pp, 2015.
- Η Γλώσσα C σε Βάθος, Νίκος Χατζηγιαννάκης, Τρίτη Έκδοση, 978-960-461-208-6, Κλειδάριθμος, 2009.

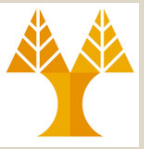
Ιστοσελίδα ΕΠΛ232

- Όλες οι πληροφορίες στο ακόλουθο URL

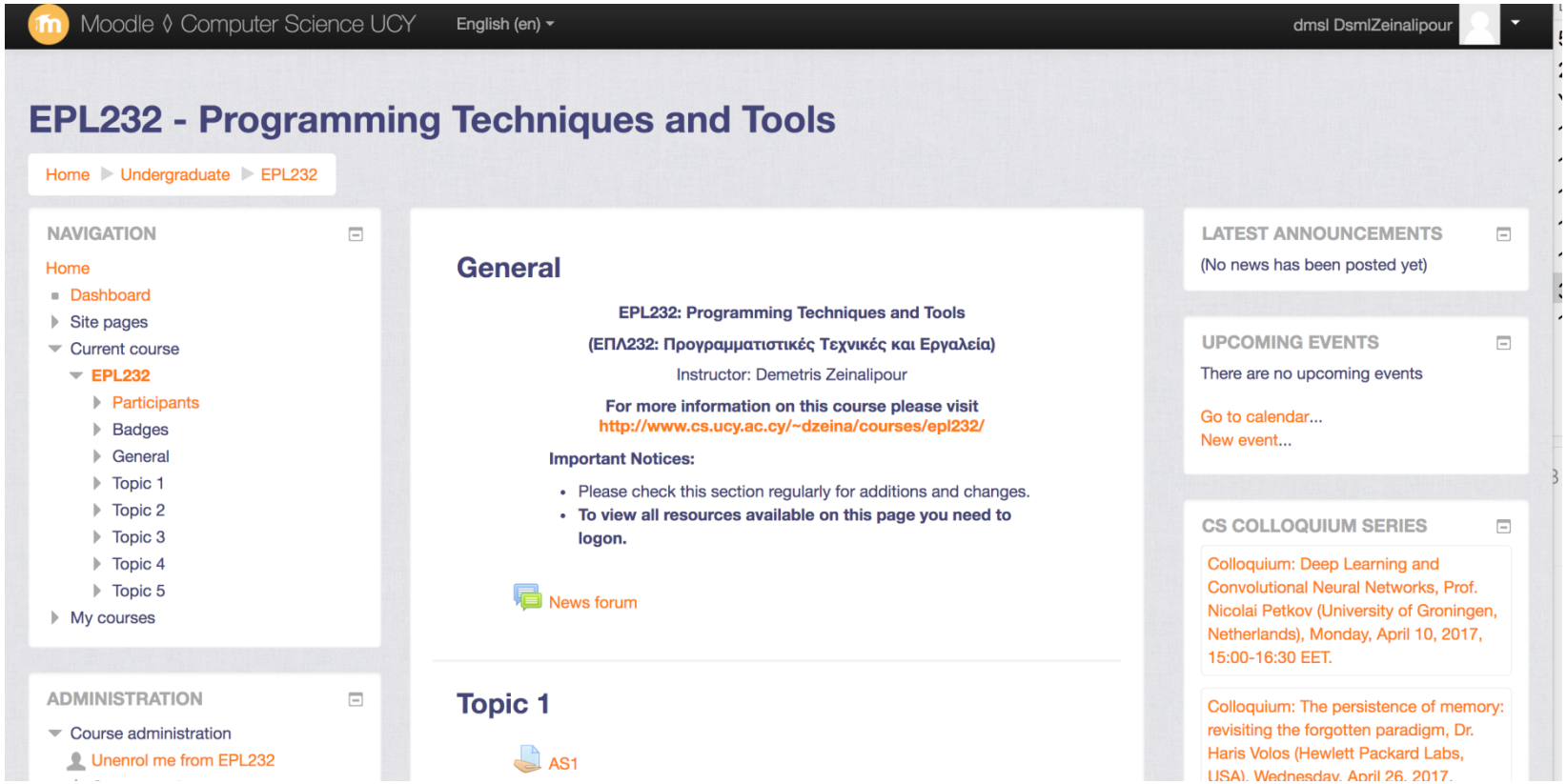
https://www.cs.ucy.ac.cy/~dzeina/courses/epl232/



Πλατφόρμα Τηλεκπαίδευσης

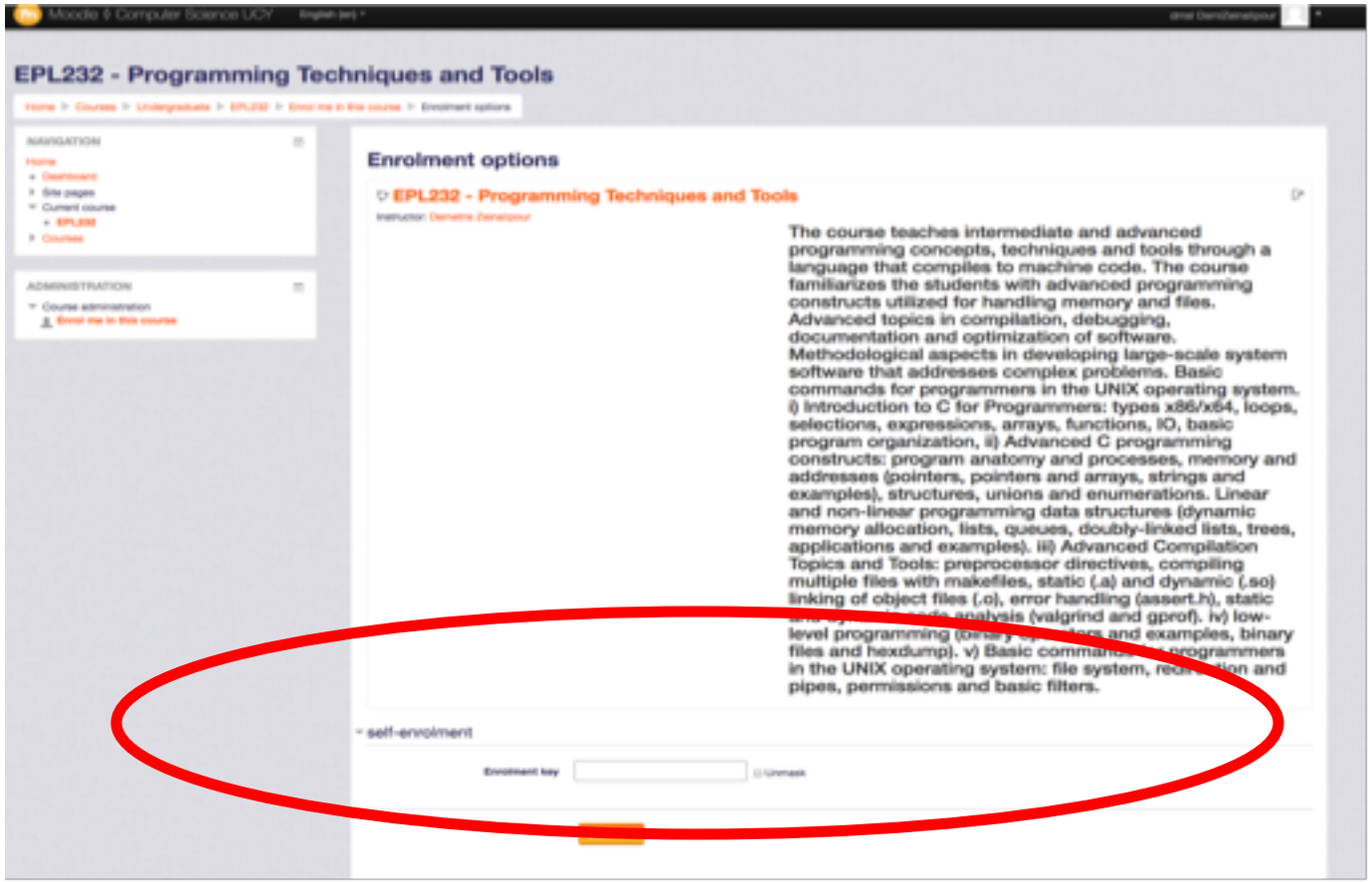


- Για τις εκπαιδευτικές δραστηριότητες του μαθήματος (**υποβολή εργασιών**, **φόρουμ** ανακοινώσεων, **ερωτηματολόγια**, **βαθμολογίες εργασιών**, κτλ) θα χρησιμοποιηθεί το Moodle: https://moodle.cs.ucy.ac.cy/



Πλατφόρμα Τηλεκπαίδευσης

Εγγραφείτε σήμερα κάνοντας χρήση του Κλειδιού Εγγραφής που θα δοθεί στην τάξη!



Αριθμός Αναζητήσεων (Turing-Complete) Γλωσσών στο WWW

Aug 2017	Aug 2016	Change	Programming Language	Ratings	Change
1	1		Java	12.961%	-6.05%
2	2		C	6.477%	-4.83%
3	3		C++	5.550%	-0.25%
4	4		C#	4.195%	-0.71%
5	5		Python	3.692%	-0.71%
6	8	^	Visual Basic .NET	2.569%	+0.05%
7	6	v	PHP	2.293%	-0.88%
8	7	v	JavaScript	2.098%	-0.61%
9	9		Perl	1.995%	-0.52%
10	12	^	Ruby	1.965%	-0.31%
11	14	^	Swift	1.825%	-0.16%
12	11	v	Delphi/Object Pascal	1.825%	-0.45%
13	13		Visual Basic	1.809%	-0.24%

Δημοτικότητα Γλωσσών Προγραμματισμού 5/9/2017

http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

The Top Programming Languages 2017 – IEEE Spectrum

C is used to write software where **speed** and **flexibility** is important, such as in **embedded systems** or **high-performance computing**.

Choose a Ranking (choose a weighting or make your own)

IEEE Spectrum | Trending | Jobs | Open | Custom

Edit Ranking | Add a Comparison |

Language Types (click to hide)

Web | Mobile | Enterprise | Embedded

Language Rank	Types	Spectrum Ranking
1. C	Mobile, Enterprise, Embedded	99.7
2. Java	Web, Mobile, Enterprise	99.4
3. C++	Mobile, Enterprise, Embedded	97.2
4. C#	Web, Mobile, Enterprise	88.6
5. JavaScript	Web, Mobile	85.5
6. Swift	Mobile, Enterprise	75.3
7. Arduino	Embedded	73.0
8. Assembly	Embedded	72.1
9. Scala	Web, Mobile	68.3
10. Haskell	Enterprise, Embedded	48.5
11. Objective-C	Mobile, Enterprise	45.2
12. Delphi	Mobile, Enterprise	41.1
13. D	Web, Embedded	38.8
14. VHDL	Embedded	36.7
15. LabView	Enterprise, Embedded	32.6

C & UNIX is Everywhere!

Vmware, Oracle, etc.

Cloud

IoT

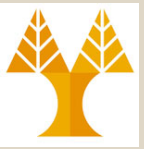
Linux Kernel (OS)

Cars

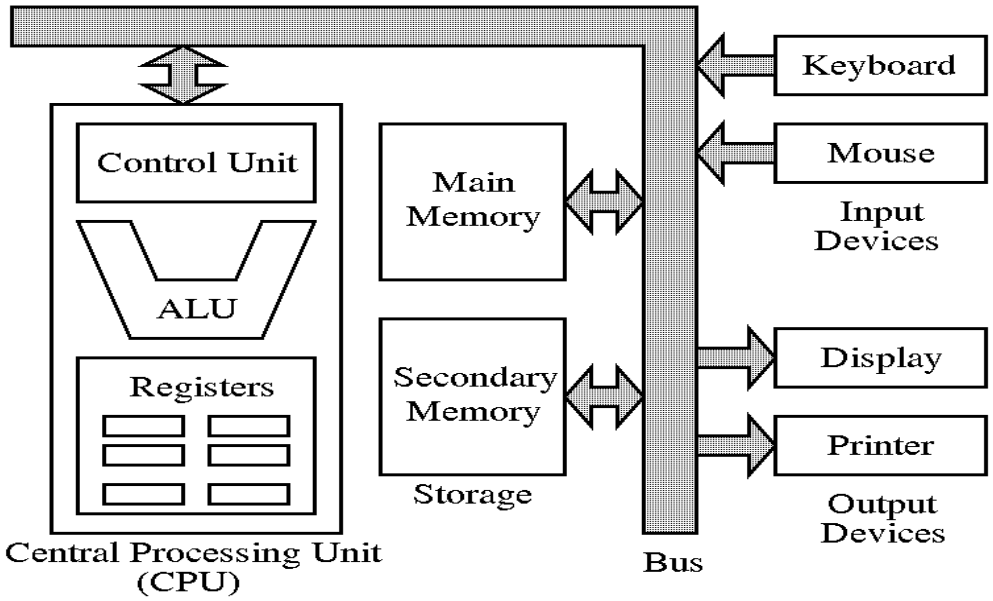
The Tesla GitHub repository contains the code for the Model S/X 2018.12 software release: **Tesla Autopilot platform**, the **kernel sources** for its underlying **hardware**, and the code for its **Nvidia Tegra-based infotainment system**.

Mobile

Γιατί μαθαίνουμε C (μετά την Java);



- **Α) Για να κατανοήσουμε σε βάθος τον κύκλο εκτέλεσης των προγραμμάτων (Διαχείριση Μνήμης, Εισόδου/Εξόδου, Δεδομένα στη Δευτερεύουσα Μνήμη, κτλ).**
  - **Η αρχιτεκτονική Von Neumann** αποτελεί το υπόβαθρο ΟΛΩΝ των σύγχρονων υπολογιστών.

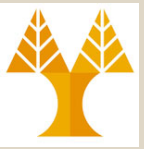


Περιγράψετε τι γίνεται όταν εκτελέσουμε ένα πρόγραμμα επεξεργασίας δεδομένων.

Γιατί μαθαίνουμε C (μετά την Java);

- Γνωρίζοντας το Von Neumann μηχάνημα, θα μπορούμε να εξηγήσουμε σε βάθος την **συμπεριφορά ενός προγράμματος και του συστήματος**
- Ιδιαίτερα, θα εκτιμήσουμε πως αλληλό-συμπληρώνονται τα μαθήματα του προγράμματος σπουδών
  - **Χαμηλού Επίπεδου / Υλικό**
    - ΕΠΛ121 Ψηφιακά Συστ. / ΕΠΛ370 Αρχιτεκ. / ΕΠΛ470 Ενσ. Συστ
    - ΕΠΛ221 Οργάνωση Υπολογιστών και Συμβολικός Προγραμματισμός
  - **Ενδιάμεσου Επιπέδου / Συστήματα**
    - Βάσεις Δεδομένων (ΕΠΛ342 και ΕΠΛ446)
    - Λειτουργικά Συστήματα (ΕΠΛ222),
    - Προγραμματισμός Συστημάτων (ΕΠΛ371)
    - Δίκτυα (ΕΠΛ324 & 375) και Ασφάλεια (ΕΠΛ475)

Γιατί μαθαίνουμε C (μετά την Java);

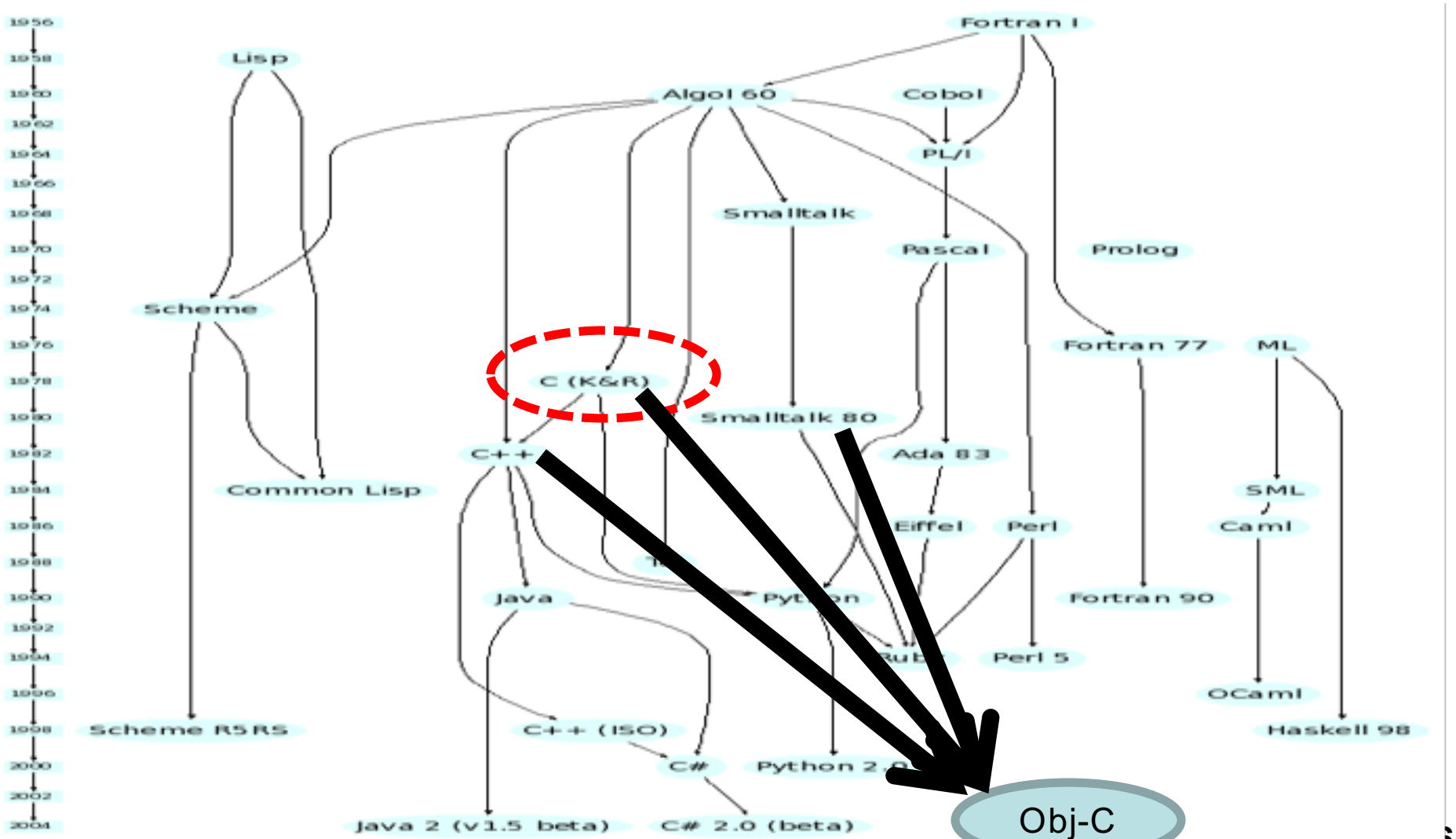


- **Β) Φιλοσοφικοί Λόγοι:** Η πολυγλωσσία είναι καλή στις μέρες μας.
- Δεν είναι όλες οι γλώσσες αντικειμενοστρεφείς….
  - **Συναρτησιακές Γλώσσες (αποκλειστική χρήση συναρτήσεων):** Haskell, Erlang, «SQL» κτλ.
  - **Λογικές Γλώσσες (αποκλειστική χρήση κανόνων και καταστάσεων):** π.χ., Prolog, Datalog Querying, R++
  - **Διαδικαστικές Γλώσσες (αποκλειστική χρήση διαδικασιών)**: C, Javascript, Fortran, Matlab, Python, Perl, Visual Basic, VB Scripting, Occam, Go, Eiffel, κτλ.
- Πολλές γλώσσες σήμερα παρέχουν διαχείριση αντικειμένων (object-based, π.χ., Obj.name), αλλά όχι αντικειμενοστρέφια.
  - Επομένως είναι καλό να εξασκηθούμε σε ένα διαφορετικό και διαδεδομένο μοντέλο προγραμματισμού.

Γιατί μαθαίνουμε C (μετά την Java);

- **Γ) Άλλοι Λόγοι:**
  - Η C είναι **αποδοτική (efficient)!**
  - Η C είναι η **γλώσσα** του **Unix/Linux,** πλατφόρμες που λειτουργούν το μεγαλύτερο ποσοστό των παγκόσμιων υποδομών & συσκευών στις μέρες μας.
  - Η C είναι **προτυποποιημένη** (standard), **φορητή** (portable), **αρθρωτή** (modular) και **επιτρεπτική (permissive).**
    - Κατάλληλη για προχωρημένους προγραμματιστές…
  - Η C είναι η βάση της C++, της Java/C#, της Obj.-C
    - Δες επόμενες διαφάνειες
  - Η C είναι **Χαμηλού Επιπέδου** και **Ψηλού Επιπέδου**
    - από GUI μέχρι Συμβολικό Κώδικα (Assembly)!

Γλώσσες Βασισμένες στη C



Γλώσσες Βασισμένες στην C

«Όταν μάθετε C όλες οι άλλες γλώσσες θα είναι εύκολες»

- ***C++*** περιλαμβάνει όλα τα πλεονεκτήματα της C, αλλά περιλαμβάνει **κλάσεις (classes)** και άλλες έννοιες για **Αντικειμενοστρεφή προγραμματισμό.**
  - π.χ., Windows Systems & Applications, Libraries, κτλ.
- ***Java*** βασίζεται στην C++ αλλά την απλοποιεί εισάγοντας **απλουστευμένη σύνταξη**, διαχειριστή μνήμης (garbage collector), κ.α.
  - Web Applets, Enterprise Progr. (DBs), **Android**, etc.
- ***C#*** είναι πλέον η πιο διαδεδομένη γλώσσα της Microsoft βασισμένη στο μοντέλο της Java.
  - Microsoft .NET εφαρμογές. **Window Phone**, Cloud, κτλ.
- **Objective-C** είναι η γλώσσα για πλατφόρμες Apple
  - Ουσιαστικά πρόκειται για C με απλή αντικειμενοστρέφια
  - Π.χ., Εφαρμογές για Apple Mac, IPhone, IPad, IPod, κτλ.

Γλώσσες Βασισμένες στην C

«Ομοιότητα σύνταξης γλωσσών με την C. Απλά άλλες γλώσσες δίνουν περισσότερες βιβλιοθήκες και ΟΟ έννοιες»

C (GNU, Cross-Platform) - Procedural

```
// hello.c
#include <stdio.h>
int main(int argc, const char *argv[] ) {
    printf( "hello world\n" );
    return 0;
}
```

C++ (GNU, Cross-Platform) - Obj. Orie.

```
// hello.cpp
#include <iostream> using namespace std;
int main () {
  cout << "Hello World!";
  return 0;
}
```

JAVA (Sun/Oracle, Cross-Platform) - OO

```
// hello.java
public class hello {
   public static void main (String args []) {
      System.out.println("Hello world");
   }
}
```

C# (Microsoft) - OO

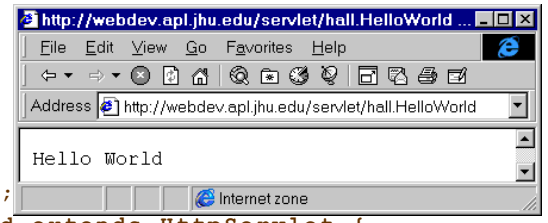
```
// Hello1.cs
public class Hello1 {
   public static void Main() {
      System.Console.WriteLine("Hello, World!");
   }
}
```

Objective-C (Apple) - Proc. ή OO

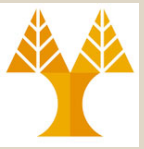
```
// hello.m
#import <stdio.h>
int main( int argc, const char *argv[] ) {
    printf( "hello world\n" );
    return 0;
}
```

JAVA (Servlet)

```
package hall;
import java.io.*;
import javax.servlet.*;
public class HelloWorld extends HttpServlet {
  public void doGet(HttpServletRequest request,
     HttpServletResponse response)
      throws ServletException, IOException {
    PrintWriter out = response.getWriter();
    out.println("Hello World");  }
}
```



Το πρώτο πρόγραμμα C

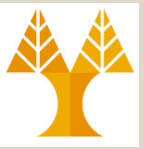


```
#include <stdio.h>

int main(void)
{
  printf("Hello World!\n");
  return 0;
}
```

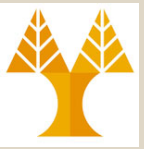
- Το πιο πάνω πρόγραμμα αποθηκεύεται σε αρχείο με όνομα `hello.c`.
- Το όνομα του αρχείου μπορεί να είναι οτιδήποτε, αλλά η κατάληξη `.c` συχνά απαιτείται από τους μεταγλωττιστές.

Μεταγλώττιση & Σύνδεση



- Πριν εκτελεστεί ένα πρόγραμμα, τρία βήματα είναι συνήθως απαραίτητα:
  - ***Προεπεξεργασία (Preprocessing).*** επεξεργασία εντολών αρχείου που ξεκινάνε με #, γνωστά ως οδηγίες προεπεξεργαστή (***directives***)
  - ***Μεταγλώττιση (Compiling).*** Μεταγλώττιση του αρχείου σε γλώσσα μηχανής (***object code***).
  - ***Σύνδεση (Linking).*** Ο ***linker*** συνθέτει το object code των επί μέρους αρχείων με οτιδήποτε επιπλέον κώδικα απαιτείται για να παραχθεί ένα εκτελέσιμο αρχείο.
- Ο preprocessor είναι συνήθως μέρος του compiler και όλα τα πιο πάνω εκτελούνται με μια όπως θα δούμε στην επόμενη διαφάνεια.

Μεταγλώττιση & Σύνδεση



- Ο μεταγλωττιστής C του UNIX είναι ο CC.

  `% cc hello.c` (όπου `%` η γραμμή εντολών UNIX)

- Το Linking γίνεται αυτόματα (εάν και μπορεί να γίνει επιλεκτικά με τη `ld` εντολή, θα το δούμε αργότερα)
- Το αποτέλεσμα είναι το `a.out` αρχείο το οποίο είναι το **εκτελέσιμο πρόγραμμα** (executable)
  - Το όρισμα `-o` επιτρέπει τον προσδιορισμό του ονόματος του εκτελέσιμου

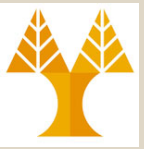
    ```
    % cc -o hello hello.c  ; ./hello
    ```

- Στο μάθημα θα χρησιμοποιήσουμε MONO **τον GNU GCC μεταγλωττιστή**

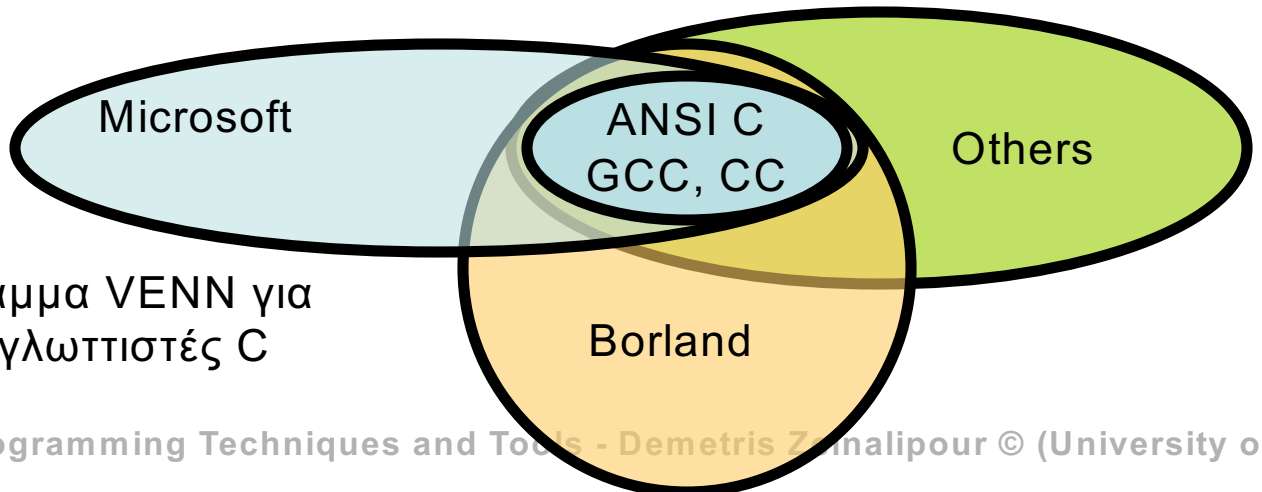
  `% gcc -o hello hello.c`

Εάν το μονοπάτι μεταγλώττισης δεν είναι στην μεταβλητή περιβάλλοντος PATH, τότε απαιτείται το «./»

Μεταγλώττιση & Σύνδεση

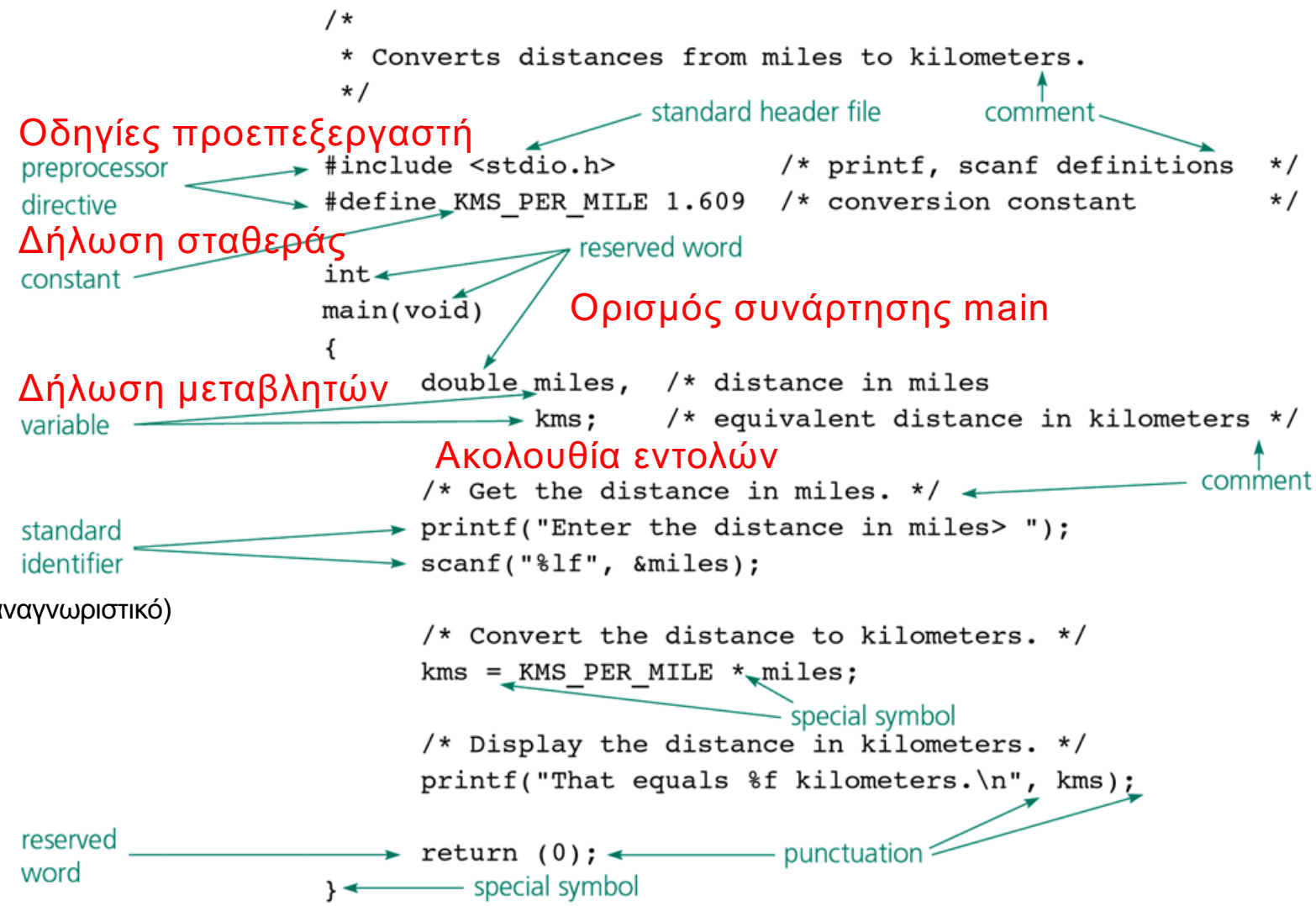


- Η C αποτελείται από ένα σύνολο συντακτικών κανόνων.
- Για να εκτελεστεί ένα πρόγραμμα C πρέπει να χρησιμοποιηθεί ένας **μεταγλωττιστής (compiler),** ο οποίος κατασκευάζεται από διάφορες εταιρείες και οργανισμούς.
- Η **American National Standard Institutes (ANSI)** δημιούργησε το πρότυπο ANSI C για λόγους μεταφερσιμότητας (portability) του κώδικα το οποίο καλούνται οι διάφορες εταιρείες να ακολουθούν.
  - Εμείς θα χρησιμοποιήσουμε τον GNU GCC ο οποίος είναι συμβατός με την ANSI C18 (2018), C11 (2011) και C99 (1999) έκδοση, ενώ αρκετοί άλλοι υποστηρίζουν μόνο C89 και άλλες προεκτάσεις εκτός προτύπου.

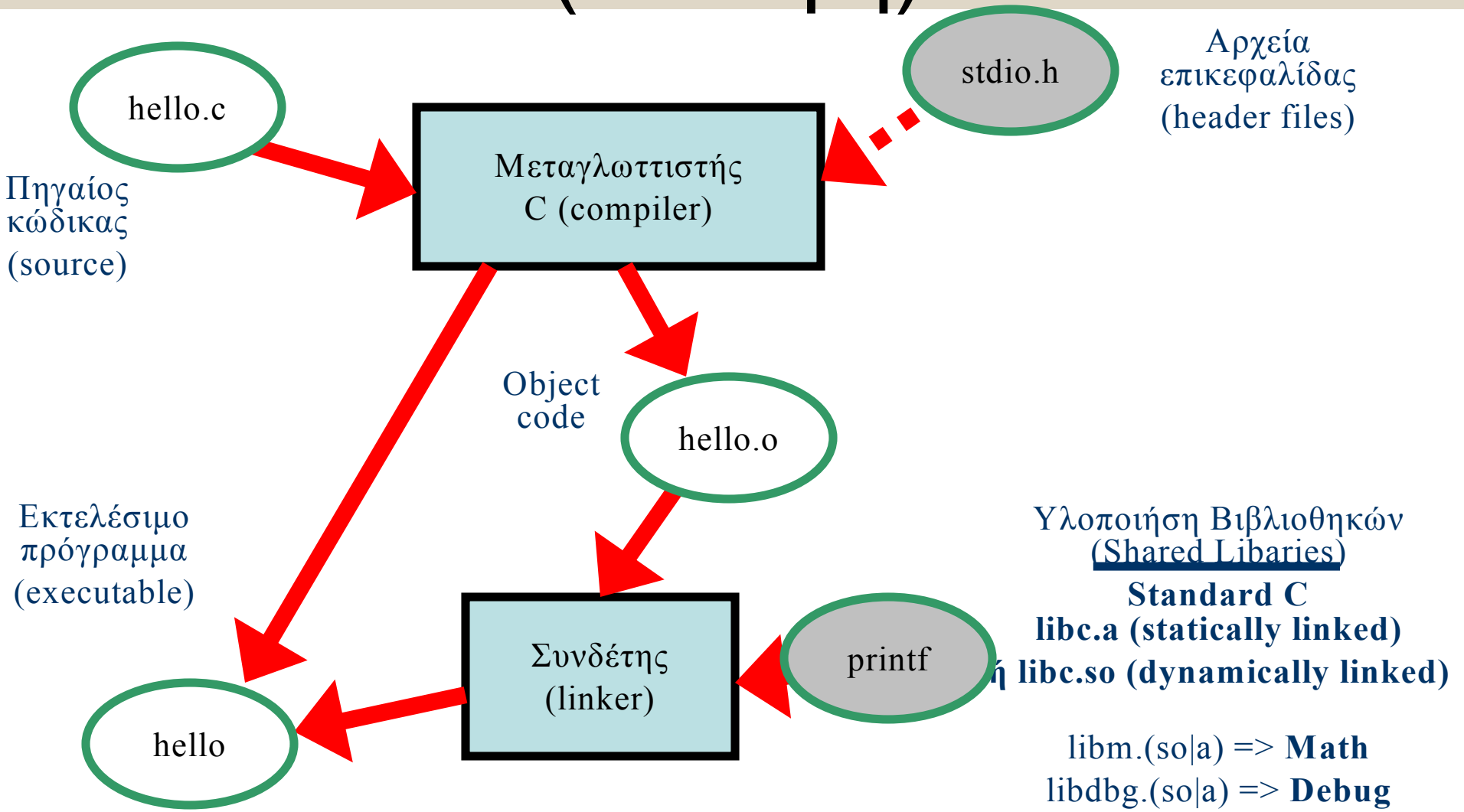


Διάγραμμα VENN για μεταγλωττιστές C

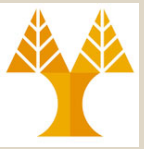
Το δεύτερο πρόγραμμα C
Εύρεση Χιλιομέτρων από Μίλια



Μεταγλώττιση με GCC (Σύνοψη)



Μεταγλώττιση με GCC

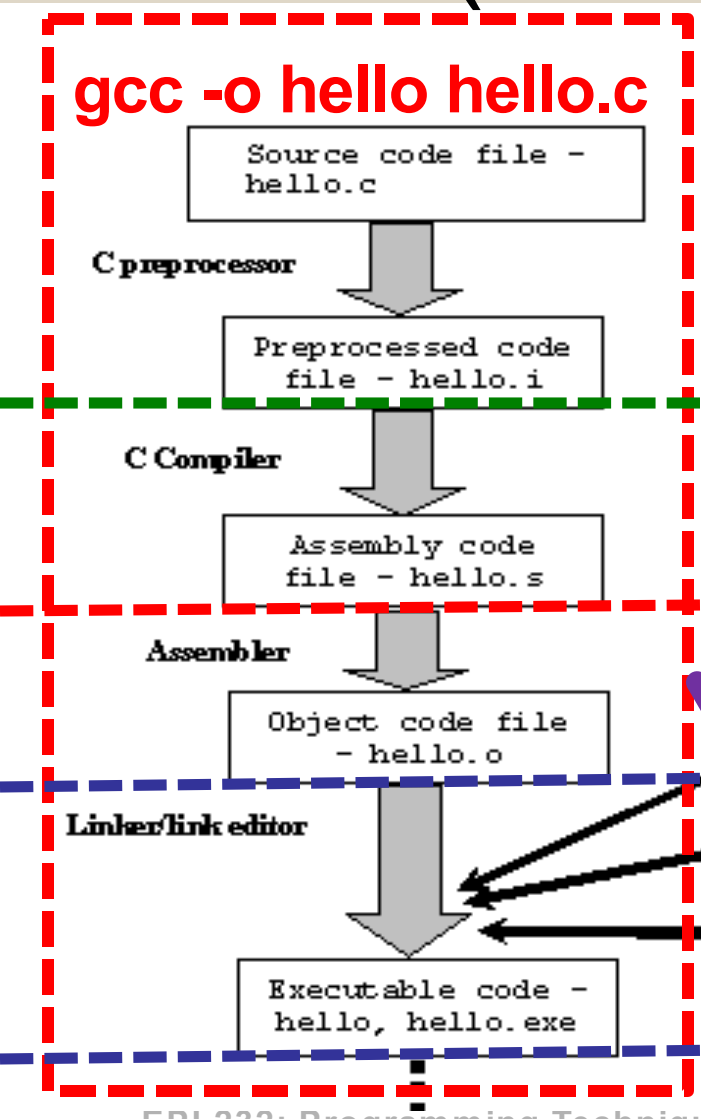


- Η συμπεριφορά του μεταγλωττιστή μπορεί να αλλάξει με εκατοντάδες ορίσματα (δείτε εντολή: man gcc).

-o *file*	output file for object or executable
-Wall -W	all warnings – use always!, -W: issues additional warnings beyond those produced by –WALL (use both anyway …)
-c	compile single module (that has no main())
-pedantic	Causes programs using nonstandard features to be rejected
-g	insert debugging code (gdb, μελλοντικά)
-p	insert profiling code (gprof, μελλοντικά)
-l	Library, π.χ., lpthread -lssl –lcrypto **-lm** (math)
-E	Stop after the preprocessing stage and output the preprocessed source code (μελλοντικά)
-O	optimization for code size and execution time (free() problems)

gcc -Wall -Wuninitialized -Wunreachable-code –pedantic file.c

Μεταγλώττιση με GCC (Αναλυτικά Βήματα)



gcc –E hello.c > hello.i

Οι ορισμοί των βιβλιοθηκών που περιλαμβάνονται στο #include, τοποθετούνται μαζί με το .c κώδικα στο .i αρχείο.

gcc –S hello.c

Παράγεται αρχείο **hello.s** που περιέχει τον συμβολικό κώδικα του προγράμματος.

gcc –c hello.c

Relocation object code information

Other objects file/modules

Library files

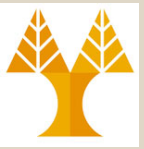
as -arch x86_64 –o hello.o hello.s

Ο συμβολομεταφραστής (assembler) AS ή GAS (GNU), μεταφράζει το συμβ. κώδικα σε κώδικα μηχανής x86_64

ld -e _main –o hello -lc hello.o

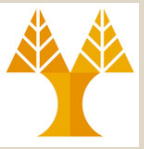
Ο αυνδέτης (linker) παράγει το τελικό εκτελέσιμο.

Σφάλματα Μεταγλώττισης



- Υπάρχουν πολλαπλοί Λόγοι
  - **Pre-Processor (**Προεπεξεργαστή, gcc -E)
    - Π.χ., #include <non-existent-library.h>
  - **Parser** (Συντακτικός Αναλυτής):
    - Π.χ., ξεχνάμε να κλείσουμε μια παρένθεση.
  - **Assembler** (Συμβολομεταφραστή, as):
    - Μετατρέπει την **συμβολική γλώσσα (assembly)** σε **αντικειμενικό κώδικα (object code)**
    - Σπάνια λάθη που σχετίζονται με τον As
  - **Linker (ld):** Σύνδεση με μη-υπαρκτή συνάρτηση

Σφάλματα Μεταγλώττισης



- Εάν ο `gcc` μπερδευτεί, τότε παρουσιάζονται εκατοντάδες μηνύματα:
  - Διορθώστε το πρώτο, μετά δοκιμάστε ξανά – αγνοώντας τα υπόλοιπα.
- Ο `gcc` θα δημιουργήσει ένα εκτελέσιμο με **προειδοποιήσεις (warnings)**, αρκεί να μην υπάρχει **λάθος (error)**
  - Μην αγνοείτε τις προειδοποιήσεις!
  - Κάνετε χρήση του `gcc -Wall` για να παρουσιάσετε ΌΛΕΣ τις προειδοποιήσεις, π.χ., :
    - `if (x = 0)` **vs.** `if (x == 0)`
    - `example.c:3: warning: suggest parentheses around assignment used as truth value`

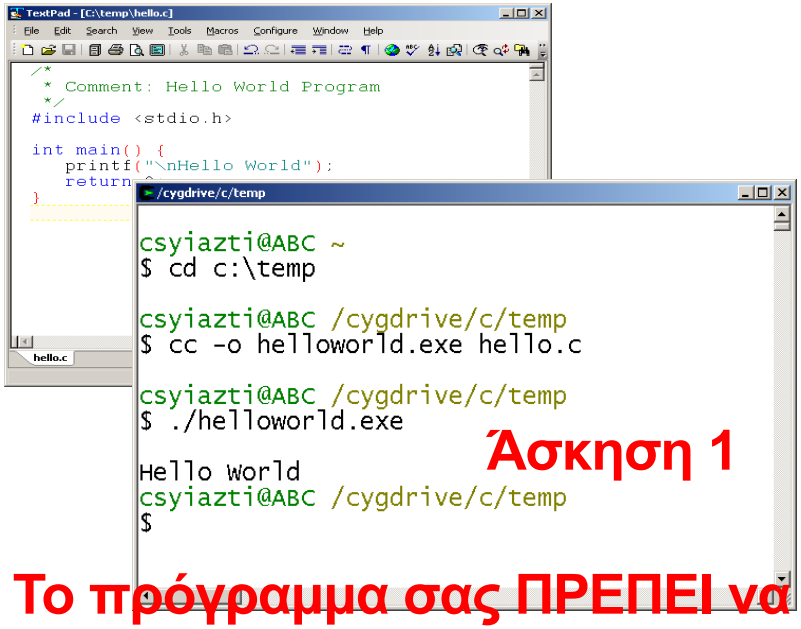
Σφάλματα Μεταγλώττισης

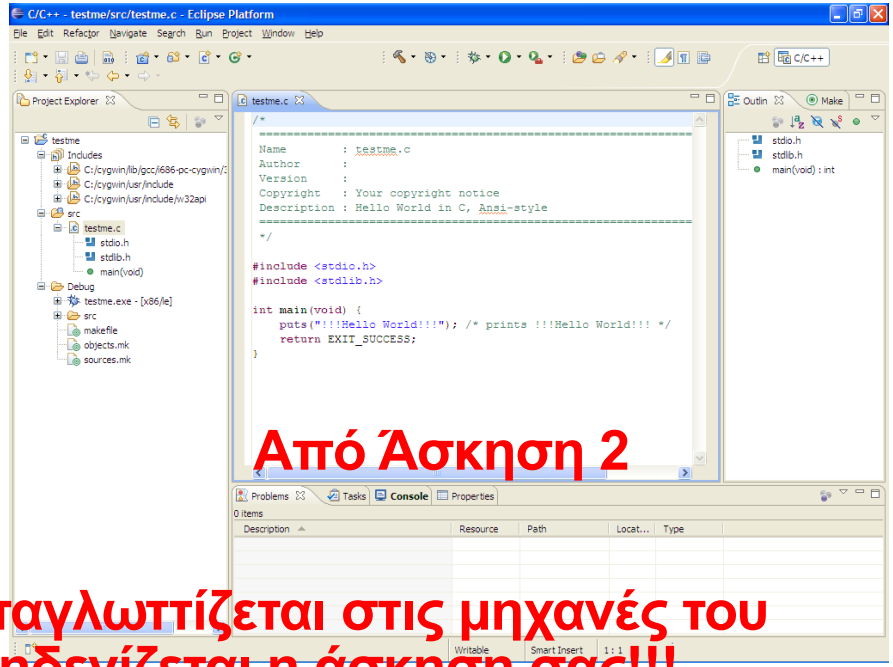
- Ο GCC δημιουργεί **object code** για κάθε αρχείο.
- Θεώρει ότι **αναφορές σε συναρτήσεις** και άλλες μεταβλητές θα **επιλυθούν αργότερα** κατά το linking.
- Εάν το **πρόγραμμα** σας **αναφέρεται** σε συνάρτηση (π.χ., print) που **δεν υπάρχει** ορισμένη στο πρόγραμμα σας θα πάρετε λάθος κατά την **σύνδεση (linking)** :

```
undefined symbol    first referenced in file
 _print             program.o
ld fatal: Symbol referencing errors
No output written to file.
```

Ολοκληρωμένα Περιβάλλοντα Ανάπτυξης (Integrated Development Environments)

- Ένα ***Integrated Development Environment (IDE)*** είναι ένα λογισμικό το οποίο επιτρέπει τη συγγραφή, μεταγλώττιση, εκτέλεση και αποσφαλμάτωση ενός προγράμματος.
- Για αρχή θα χρησιμοποιήσουμε τον συνδυασμό **κελύφους με κάποιο κειμενογράφο** για προγραμματιστές και στη συνέχεια του μαθήματος θα **λειτουργούμε μόνο με το IDE**.





Το πρόγραμμα σας ΠΡΕΠΕΙ να μεταγλωττίζεται στις μηχανές του εργαστηρίου εναλλακτικά θα μηδενίζεται η άσκηση σας!!!

Native vs. Intermed. Compile [C/C++/Objective-C vs. JAVA/Android/C#]

- **C/C++/Objective-C(iOS) :** Ο πηγαίος κώδικας μετατρέπεται πλήρως σε γλώσσα μηχανής κατά την μεταγλώττιση.
- **JAVA, Android, C#:** Ο πηγαίος κώδικας μετατρέπεται σε **ενδιάμεσο κώδικα (bytecode)**, ο οποίος μπορεί να εκτελεστεί σε οπουδήποτε σύστημα έχει το λεγόμενο **Java Virtual Machine (JVM).**
- Αυτό που γίνεται πρακτικά στο **JAVA, Android, C#** είναι ότι το bytecode μετατρέπεται κατά την εκτέλεση του προγράμματος σε κώδικα μηχανής από το (**Just-in-time JIT Compilation)** υποσύστημα του JVM.
  - Στη **Java** JIT γίνεται από το JVM (**Java Virtual Machine**)
  - Στη **C#** JIT γίνεται από το CLR (**Common Language Runtime**)
  - Στο **Android** JIT γίνεται από το DVM (**Dalvik Virtual Machine**), ή το **ART (Android RunTime)** σε εκδόσεις του Android μετά το 4.4 (Έκδοση 8 το 2017).
- Στη JAVA 9 (έκδοση 2017) υπάρχει το **ahead-of-time (AoT)** compilation, όπου η μεταγλώττιση γίνεται at runttime αλλά πριν τη χρήση για λόγους επίδοσης.

Σχόλια

- ***/*from to comment*/***
- // C99 και C++-style σχόλιο!
- Σύμβαση για μακρύτερα σχόλια:

```
/*      Καλύτερη ευκρίνεια  ότι υπάρχει σχόλιο σε αυτή τη γραμμή
 * AverageGrade()
 * Given an array of grades, compute the average.
 */
```

Θα υπάρχει εξειδικευμένο εργαστήριο για συστάσεις πάνω σε Στυλ γραφής σχολίων και κώδικα!

- Αποφεύγετε την χρήση περιέργων κουτιών ****
  - Το βιβλίο εισηγείται το ακόλουθο πλάτους 60 χαρακ.

```
/*********************************************************
 * print_result: Notifies the user of the result, using  *
 *               the external variables set by           *
 *               analyze_hand.                           *
 *********************************************************/
```

  - Επίσης, μη χρησιμοποιείτε ΠΟΤΕ TAB αλλά 3 SPACES στη θέση κάθε TAB εφόσον αυτά δεν αλλάζουν πλάτος μεταξύ κειμενογράφων

Αριθμητικοί Τύποι Δεδομένων (32/64-bit Προγραμ. Μοντέλα)

- Λίγα λόγια για τύπους δεδομένων με τους 2 πιο διαδεδομένα μοντέλα δεδομένων
  - **I**(ntegers) L(ong) P(ointer) 32 => x86 Model
  - L(ong) P(ointer) 64 => x64 Model

Datatype	**ILP32 Model**	**LP64 Model**
char	8	8
short	16	16
int	**32**	32
long	**32 (4 bytes)**	**64 (8 bytes)**
pointer	**32 (4 bytes)**	**64 (8 bytes)**

Η μνήμη μπορεί να έχει ΜΟΝΟ μέχρι $2^{32} = \sim 4 \times 10^9$ (δηλ., 4GB) διευθύνσεις ☹!

Η μνήμη μπορεί να έχει μέχρι 16 Exa ($\times 10^{18}$) διευθύνσεις ☺!

Δοκιμάστε στο σπίτι το ακόλουθο:
printf("%d, %d, %d", sizeof(int), sizeof(long), sizeof(void *));

Εκτυπώνοντας μια Μεταβλητή

- **`printf("%d", variable)`**
  - `%d` integer, `%ld` (64 bit), `%x` hex., `%o` octal
  - `%f` float (6 δεκ. ψηφ.), `%.2f` (2 δεκ. ψηφ.)
  - `%c` char
  - `%s` string (πίνακας char με τελικό NUL (\0))
- Υπάρχουν εκατοντάδες ορίσματα κάποια εκ' των οποίων θα δούμε συνοπτικά στην ερχόμενη διάλεξη.
  - το Κεφ. 22 καλύπτει το θέμα σε περισσότερο βάθος το οποίο δε θα χρειαστεί για αυτό το μάθημα.

Διαβάζοντας μια Μεταβλητή

- **`scanf("%f", &x);`**
  - `%d` integer, `%ld` (64 bit), `%x` hex., `%o` octal
  - `%f` float
  - `%c` char
  - `%s` string (πίνακας char με τελικό NUL (\0))
- Και πάλι, υπάρχουν εκατοντάδες ορίσματα κάποια εκ' των οποίων θα δούμε συνοπτικά στην ερχόμενη διάλεξη.
  - το Κεφ. 22 καλύπτει το θέμα σε περισσότερο βάθος το οποίο δε θα χρειαστεί για αυτό το μάθημα.

Αναγνωριστικά της C (Identifiers)

- Τα ονόματα **μεταβλητών**, **συναρτήσεων**, **μακρο-εντολών** και **οντοτήτων** ονομάζονται **αναγνωριστικά (*identifiers*).**
  - Η C όπως και η JAVA και C# (αντίθετα με π.χ., VB) είναι **ευαίσθητη στο τύπο χαρακτήρα (case sensitive)**
  - Ένα αναγνωριστικό μπορεί να περιέχει **letters**, **digits**, και **underscores**, αλλά πρέπει πάντα να ξεκινά με **letter** ή **underscore**:

```
times10   get_next_char   _done   OK
10times   get-next-char           ERROR
```

- Επιβάλλεται ένα ομοιόμορφο στυλ, π.χ.,
  - `symbol_table` ή `symbolTable`
  - 3 κενά spaces ανά εμφωλευμένη έκφραση (όχι tabs)
  - 80 στήλες ΜΟΝΟ. (περισσότερα στο εργαστήριο 3)

A word on case

- ~~Sentence Case~~
- UPPERCASE
- lowercase
- camelCase
- ***PascalCase***
- hypen-case
- snake_case

Δεσμευμένα Αναγνωριστικά

- Οι ακόλουθες **λέξεις κλειδιά** (***keywords)*** ΔΕΝ μπορούν να χρησιμοποιηθούν για αναγνωριστικά στην C:

```
auto        enum        restrict*   unsigned
break       extern      return      void
case        float       short       volatile
char        for         signed      while
const       goto        sizeof      _Bool*
continue    if          static      _Complex*
default     inline*     struct      _Imaginary*
do          int         switch
double      long        typedef
else        register    union
```

***C99 only**