

ΜΥΥ502

Προγραμματισμός Συστημάτων



Β. Δημακόπουλος

dimako@cse.uoi.gr

<http://www.cse.uoi.gr/~dimako>

❖ Αντικείμενο μαθήματος:

- Εκμάθηση βασικών εργαλείων, τεχνικών και μεθόδων για προχωρημένο προγραμματισμό που στοχεύει περισσότερο στο «σύστημα» παρά στην «εφαρμογή»
- Η γλώσσα C είναι μονόδρομος στον προγραμματισμό συστημάτων
 - ✧ Με μεγάλη προγραμματιστική βάση και στο χώρο των εφαρμογών
 - ✧ Προγραμματισμός συστήματος:
 - «κάτω» από το επίπεδο εφαρμογών (υποστηρίζει τις εφαρμογές)
 - Ανάμεσα στις εφαρμογές και το hardware
 - π.χ. μεταφραστές, λειτουργικά συστήματα, συστήματα υποστήριξης εκτέλεσης, ενσωματωμένα συστήματα κλπ.
- Το UNIX/POSIX και τα «POSIX-οειδή» περιβάλλοντα είναι η σημαντικότερη και πιο ολοκληρωμένη πλατφόρμα για εργασία σε επίπεδο συστήματος
 - ✧ Βασικές γνώσεις χρήσιμες και σε επόμενα μαθήματα (π.χ. λειτουργικά συστήματα, παράλληλα συστήματα, μεταφραστές κλπ)

❖ Ύλη μαθήματος (από τον οδηγό σπουδών):

- “ Η γλώσσα προγραμματισμού C: στοιχειώδης C (βασικοί τύποι δεδομένων, εκφράσεις, τελεστές, δομές ελέγχου ροής, συναρτήσεις), προχωρημένα στοιχεία (πίνακες, δείκτες, δομές), δυναμική διαχείριση μνήμης, είσοδος/έξοδος, προεπεξεργαστής. ”
- “ Βασικές κλήσεις UNIX (διεργασίες, I/O, σήματα). Διαδιεργασιακή επικοινωνία (κοινόχρηστη μνήμη, sockets). Εισαγωγή στον παράλληλο προγραμματισμό (νήματα, mapReduce). Προχωρημένα θέματα (ασφάλεια, γλώσσα μηχανής, εργαλεία ανάπτυξης μεγάλων προγραμμάτων). ”

❖ Δύο μέρη:

➤ Γλώσσα προγραμματισμού C

- ✧ Υποθέτει γνώση προγραμματιστικών τεχνικών
- ✧ Υποθέτει γνώση γλωσσών προγραμματισμού «συγγενών» με την C (π.χ. Java)
- ✧ Καλύπτονται από τις «Τεχνικές Αντικειμενοστραφούς Προγραμματισμού» και «Ανάπτυξη Λογισμικού»

Π1: Περίπου 50% της ύλης

➤ Προγραμματισμός συστημάτων POSIX και προχωρημένα θέματα

- ✧ Εμβάθυνση σε προχωρημένες δυνατότητες της C
- ✧ Γνωριμία με διαδικασίες και εργαλεία ανάπτυξης εφαρμογών συστήματος
- ✧ Βασικές κλήσεις POSIX (διεργασίες, σήματα, επικοινωνίες, νήματα κλπ)
- ✧ Άλλα προχωρημένα θέματα και τεχνικές

Π2: Περίπου 50% της ύλης

- ❖ Υπάρχουν πολλά βιβλία για C
 - Και πάρα πολύ υλικό στο διαδίκτυο
- ❖ Για προγραμματισμό συστημάτων (POSIX) όχι τόσα πολλά μεταφρασμένα
 - Συνήθως θεωρούν δεδομένη τη γνώση της C
 - Πολλά που είναι για χρήση / διαχείριση του UNIX, όχι προγραμματισμό
 - ✧ Δεν μας αφορούν

Προγραμματισμός σε UNIX

ΔΕΥΤΕΡΗ ΑΜΕΡΙΚΑΝΙΚΗ ΕΚΔΟΣΗ



MARC J. ROCHKIND

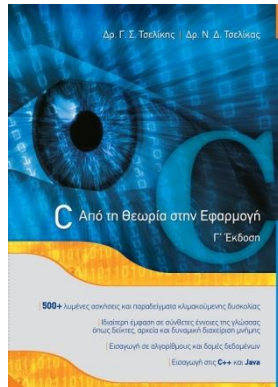
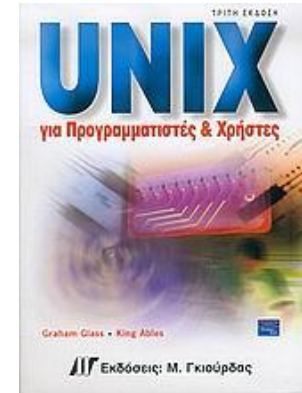


➤ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΕ UNIX, M.J. Rochkind (2007)

- ✧ Εξαιρετικό βιβλίο για προγραμματισμό συστημάτων UNIX
- ✧ Υποθέτει γνώση της C

➤ UNIX ΓΙΑ ΠΡΟΓΡΑΜΜΑΤΙΣΤΕΣ ΚΑΙ ΧΡΗΣΤΕΣ, G.Glass, K. Ables (2005)

- ✧ Χρήση, διαχείριση, εσωτερικά του UNIX
- ✧ Δύο κεφάλαια αφιερώνονται στα εργαλεία προγραμματισμού και στις κλήσεις συστήματος του UNIX και υποθέτει γνώση της C

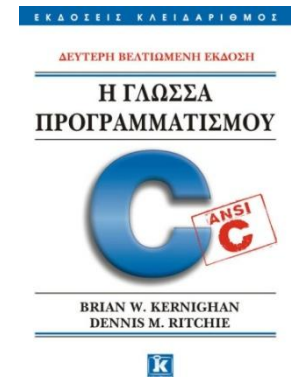


➤ C ΑΠΟ ΤΗ ΘΕΩΡΙΑ ΣΤΗΝ ΠΡΑΞΗ, Γ. Τσελίκης, Ν. Τσελίκης (2016)

- ✧ Πάρα πολύ καλό βιβλίο C (ελληνικό!)

➤ Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C, B.W. Kernighan, D.M. Ritchie (2008)

- ✧ «Ευαγγέλιο» της C (C90)
- ✧ Ίσως όχι το καλύτερο για εκμάθηση.



- ❖ Πολλές ιστοσελίδες για C
- ❖ Πολλές ιστοσελίδες για προγραμματισμό σε UNIX/Linux/POSIX κλπ.
- ❖ Πολύ καλό βοήθημα στο διάβασμά σας:
 - “Programming in C; Unix System Calls and Subroutines using C.”, A. D. Marshall
 - <http://www.cs.cf.ac.uk/Dave/C/CE.html>
 - Καλύπτει και εκμάθηση της C αλλά και αρκετό προγραμματισμό συστημάτων POSIX

❖ Διαλέξεις:

- Δευτέρα: 12:00 – 14:00
- Τετάρτη: 12:00 – 14:00 (*)
- Αίθουσα: I5

❖ Εργαστήρια:

- Τρίτη: 14:00 – 18:00 (20:00)
- ΠΕΠ Ι, ΠΕΠ ΙΙ, ΠΕΛΣ

❖ Ώρες γραφείου διδάσκοντα:

- Τρίτη: 09:00 – 11:00
- B33

	Δε	Τρ	Τε	Πε	Πα
08:00					
09:00		Γρ			
10:00					
11:00					
12:00	Δ		Δ		
13:00					
14:00		Εργ			
15:00					
16:00		Εργ			
17:00					
18:00		Εργ			
19:00					

Ιστοσελίδα μαθήματος:

<http://www.cse.uoi.gr/~dimako/teaching/>

❖ Σκοπός/λειτουργία εργαστηρίων:

- «Φροντιστηριακού» τύπου για ενίσχυση διδασκαλίας
- Σχεδιασμός προγράμματος και υλοποίηση «επί τόπου»
- (Πολύ) έμπειροι μεταπτυχιακοί και προσωπικό για να σας βοηθήσουν

❖ Οργανωτικά:

- Ξεκινούν (λογικά) σε 2 εβδομάδες – θα σας ενημερώσω εγκαίρως
- Θα εργάζεστε δύο σε κάθε θέση (λόγω χώρου)
- 14:00 – 18:00 (2 σειρές των 2 ωρών)
- Σχεδόν κάθε εβδομάδα
- Παρουσίες

❖ Βαθμός εργαστηρίου:

- Θα βγει από τη συμμετοχή σας και από 2 προόδους
- **20% οι παρουσίες + 40% πρόοδος1 + 40% πρόοδος2**
- Κάθε απουσία σας αφαιρεί 10%. Από 2 και μετά δεν έχει διαφορά...

**Όσοι ΠΕΡΑΣΟΥΝ το
εργαστήριο, το
κατοχυρώνουν για
πάντα**

Και έρχονται μόνο στις
τελικές εξετάσεις.

Τι σημαίνει «ΠΕΡΝΑΩ» το εργαστήριο

- ❖ Για να **ΠΕΡΑΣΕΙ** κάποιος το εργαστήριο, θα πρέπει ο βαθμός του (20% οι παρουσίες + 40% πρόοδος1 + 40% πρόοδος2) να είναι $\geq 4,5$.
 - Όποιος το περάσει, το κατοχυρώνει για ΠΑΝΤΑ.

FAQ

- ❖ ΕΡΩΤΗΣΗ: *Αν κάνω όλες τις παρουσίες περνάω;*
- ❖ ΑΠΑΝΤΗΣΗ: ΟΧΙ, πρέπει να περάσετε και τις προόδους

- ❖ ΕΡΩΤΗΣΗ: *Αν ΔΕΝ περάσω για κάποιο λόγο, μπορώ να ξαναπαρακολουθήσω το εργαστήριο του χρόνου;*
- ❖ ΑΠΑΝΤΗΣΗ: ΟΧΙ, δείτε την επόμενη διαφάνεια

- ❖ ΕΡΩΤΗΣΗ: *Δεν με βολεύει η ημ/νια της προόδου / έχω κανονίσει να λείπω / έχω πληρώσει εισιτήρια / θα είμαι άρρωστος εκείνη την ημέρα. Μπορώ να δώσω κάποια άλλη μέρα;*
- ❖ ΑΠΑΝΤΗΣΗ: ΟΧΙ

❖ Αφορά μόνο όσους:

1. Έχουν ξαναεγγραφεί στο μάθημα κατά το παρελθόν ΚΑΙ
2. Δεν έχουν περάσει το εργαστήριο ποτέ

❖ Εγγραφή στα εργαστήρια:

- Υποχρεωτική, πλήρης (ηλεκτρονική) εγγραφή όπως όλοι
- **ΑΛΛΙΩΣ ΔΕΝ ΘΑ ΜΠΟΡΕΣΟΥΝ ΝΑ ΕΞΕΤΑΣΤΟΥΝ** λόγω χώρου

❖ Εξέταση και βαθμός εργαστηρίων:

- Εξέταση στις 2 προόδους του εργαστηρίου, όπως όλοι.
- Απαραίτητη επικοινωνία με διδάσκοντα πριν από κάθε πρόοδο
- Ο βαθμός θα βγει ως εξής:
50% πρόοδος1 + 50% πρόοδος2

Εργαστήρια (για όσους είχαν ΕΓΓΡΑΦΕΙ ΠΑΛΑΙΟΤΕΡΑ)

- ❖ Για να **ΠΕΡΑΣΕΙ** κάποιος παλιός το εργαστήριο, θα πρέπει ο βαθμός του (50% πρόοδος1 + 50% πρόοδος2) να είναι $\geq 4,5$.

FAQ

- ❖ ΕΡΩΤΗΣΗ: *Είχα κάνει παρουσίες παλιά. Μπορώ να πάρω το 20% από αυτές;*
- ❖ ΑΠΑΝΤΗΣΗ: **ΟΧΙ**, οι παρουσίες μετρούν μόνο την 1^η φορά που εγγράφεστε στο μάθημα

- ❖ ΕΡΩΤΗΣΗ: *Μπορώ να ξαναπαρακολουθήσω το εργαστήριο να πάρω το 20% από τις παρουσίες;*
- ❖ ΑΠΑΝΤΗΣΗ: **ΟΧΙ**

- ❖ ΕΡΩΤΗΣΗ: *Μπορώ να ξαναπαρακολουθήσω το εργαστήριο ΧΩΡΙΣ να πάρω το 20% από τις παρουσίες;*
- ❖ ΑΠΑΝΤΗΣΗ: Πολύ δύσκολο, και μόνο αν υπάρχουν ελεύθερες θέσεις. Αν όμως θέλετε να ασχοληθείτε μόνοι σας θα υπάρχουν και οι ασκήσεις και οι απαντήσεις και οι βοηθοί.

❖ Επιτυχία στο μάθημα προϋποθέτει:

1. Επιτυχία στο εργαστήριο (Βαθμός εργαστηρίων $\geq 4,5$)
 - ✧ Όσοι δεν επιτύχουν στο εργαστήριο, δεν έχουν δικαίωμα εξετάσεων
2. Τουλάχιστον βαθμό 4,5 στις εξετάσεις
 - ✧ Όσοι έχουν βαθμό < 4 στις εξετάσεις, δεν περνούν ακόμα και άριστα να πήγαν στο εργαστήριο.

❖ Τελικός βαθμός (το πιθανότερο):

- **40% εργαστήριο + 60% τελικές εξετάσεις**

Το σημερινό μάθημα

Εισαγωγικά στοιχεία για τη C



❖ D. Ritchie, Bell Labs, 1972

- Με βάση προηγούμενη γλώσσα (B)
- Χρησιμοποιήθηκε για την υλοποίηση του λειτουργικού συστήματος UNIX
- Ευρεία διάδοση από τότε.

❖ Από αυτήν προέκυψαν / επηρεάστηκαν οι περισσότερες από τις πιο δημοφιλείς γλώσσες:

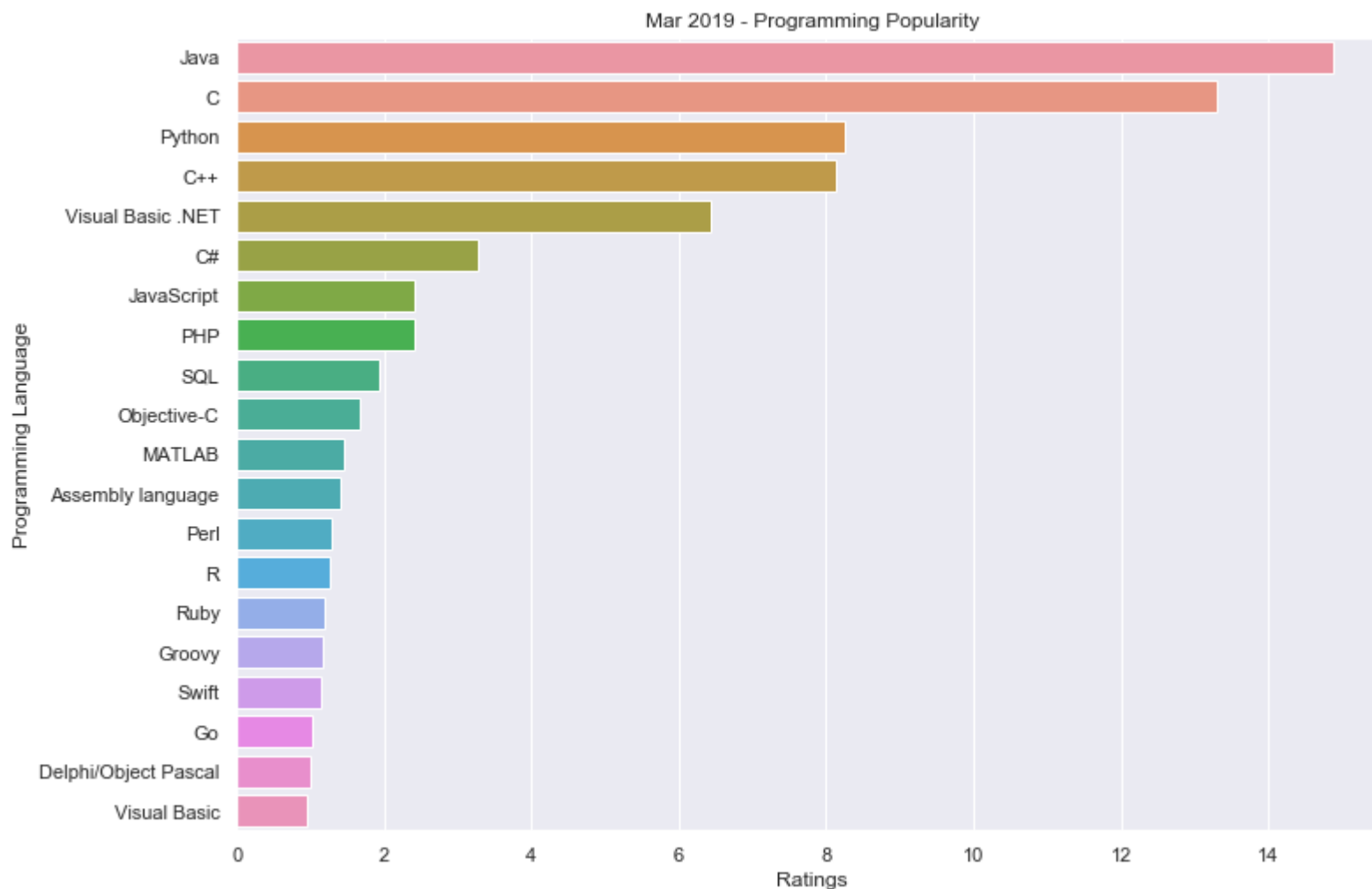
- Π.χ. C++, Java, PHP κλπ.

❖ Σε κάποιους τομείς (π.χ. ενσωματωμένα συστήματα) η C είναι ουσιαστικά η μοναδική επιλογή

- Δυνατή, μικρή, εύκολα μεταφράσιμη γλώσσα

Δημοτικότητα της C (TIOBE index, 2019)























❖ Όλες οι κορυφαίες με βάση την C!



Κατάταξη γλωσσών προγραμματισμού (IEEE Spectrum)

❖ IEEE, Αυγ. 2016

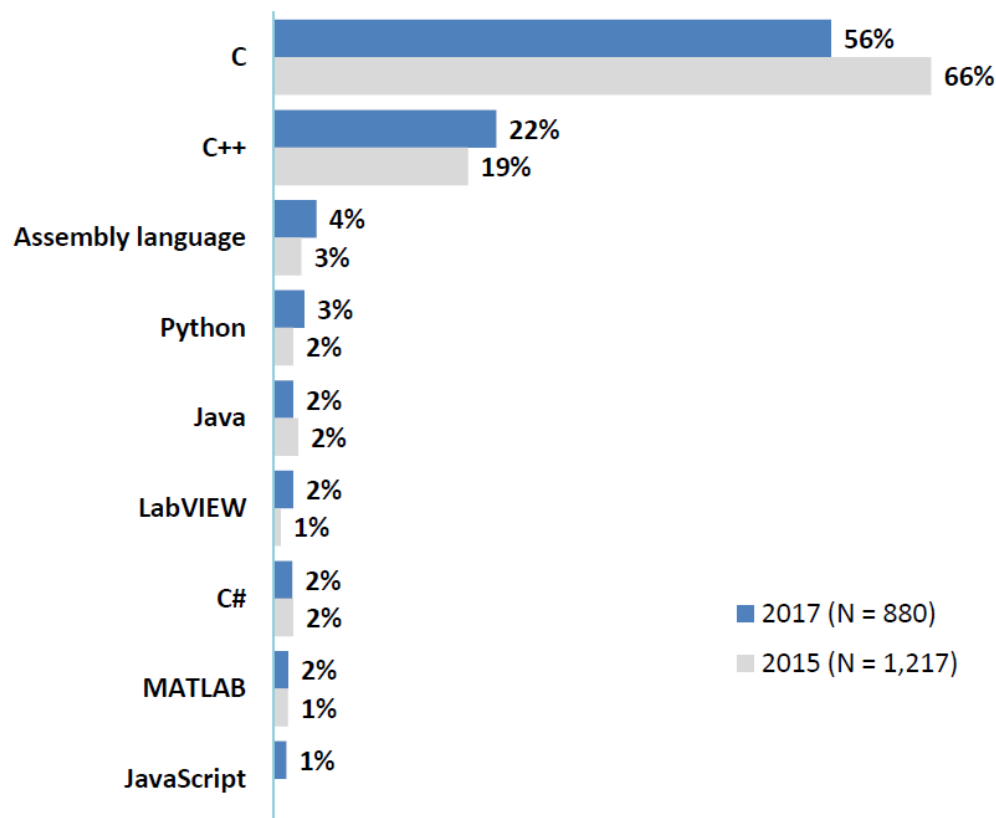
<http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages>

Language Rank	Types	Spectrum Ranking
1. C	  	100.0
2. Java	  	98.1
3. Python	 	98.0
4. C++	  	95.9
5. R		87.9
6. C#	  	86.7
7. PHP		82.8
8. JavaScript	 	82.2
9. Ruby	 	74.5
10. Go	 	71.9

My current embedded project is programmed mostly in:



ASPENCORE



- ❖ Το UNIX γράφτηκε σε C
- ❖ Ο πυρήνας του Linux είναι γραμμένος σε C
- ❖ Σχεδόν όλες οι εφαρμογές συστήματος είναι σε C
- ❖ Η πλειονότητα εφαρμογών ανοιχτού κώδικα είναι σε C
- ❖ ...

❖ Προσοχή: Η C ΔΕΝ είναι πανάκεια...

- «Επικίνδυνη» αν κάποιος δεν την ξέρει καλά
- «Χαμηλότερου» επιπέδου από άλλες γλώσσες (π.χ. Java)
- Δεν βολεύει πάντα για εφαρμογές χρήστη, ειδικά όταν υπάρχει γραφική αλληλεπίδραση

Εισαγωγή στη C

C για προγραμματιστές Java



Hello world

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

```
#include <stdio.h>

int main() {
    puts("Hello world");
    return 0;
}
```

❖ Κλάσεις

- Μόνο δεδομένα (μεταβλητές) και συναρτήσεις
- Η συνάρτηση `main()` είναι αυτή που εκτελείται αρχικά

❖ Boolean

- Με ακεραίους «προσομοιώνουμε» τα boolean
- Το **0** θεωρείται FALSE
- Οτιδήποτε μη-μηδενικό θεωρείται TRUE

❖ Strings (τουλάχιστον όπως τα χειρίζεται η Java)

- Χειρισμός μέσω πινάκων και δεικτών

❖ try ... catch μπλοκ (exceptions)

- Δεν υπάρχει ανάλογο, μόνο μέσω συναρτήσεων συστήματος

❖ Pointers (δείκτες)!

- Όχι μόνο απλό πέρασμα με αναφορά

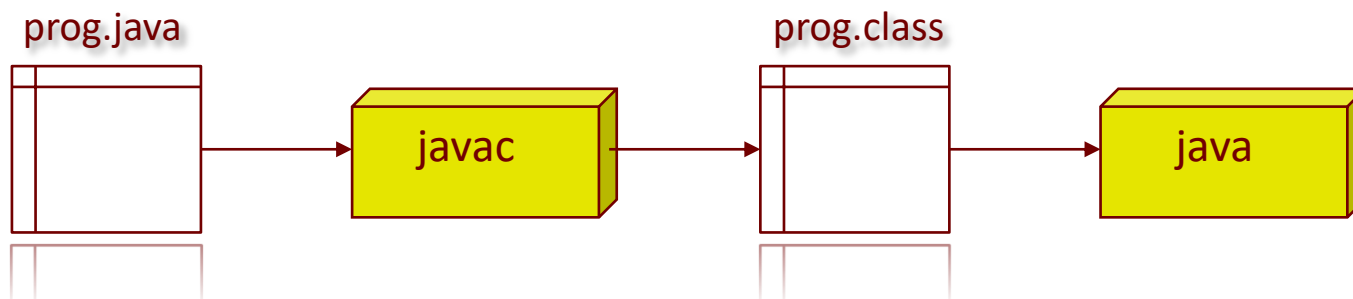
❖ «Ελευθερία» στους τύπους των δεδομένων (π.χ. int/short/char είναι πάνω-κάτω ίδιοι) και δεν γίνεται πλήρης έλεγχος κατά τη χρήση τους.

❖ «Ελευθερία» στη διαχείριση της μνήμης

- Επαφίεται πλήρως στον προγραμματιστή
- Η java έχει garbage collector που αυτόματα αποδεσμεύει άχρηστη μνήμη

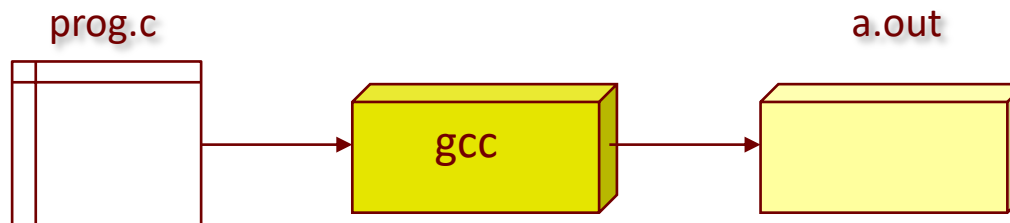
❖ Η Java είναι (basικά) ερμηνευόμενη (*interpreted*)

- Συνήθως μετατρέπεται (javac) σε bytecode,
 - ❖ ο οποίος ερμηνεύεται από μία εικονική μηχανή (JVM),
 - η οποία εκτελείται (java) στην πραγματική μηχανή



❖ Η C είναι (basικά) μεταφραζόμενη (*compiled*)

- Μετατρέπεται απευθείας σε εντολές assembly της πραγματικής μηχανής που θα εκτελέσει το πρόγραμμα
 - ❖ Το πρόγραμμα εκτελείται αυτόνομα



Το πρώτο πρόγραμμα σε C (hello.c)

```
#include <stdio.h>
int main()
{
    /* Just show a simple message */
    printf("Hello, World\n");
}
```

← HEADER (αρχείο επικεφαλίδων)
Θυμίζει το **import** της java

← Συνάρτηση εκκίνησης

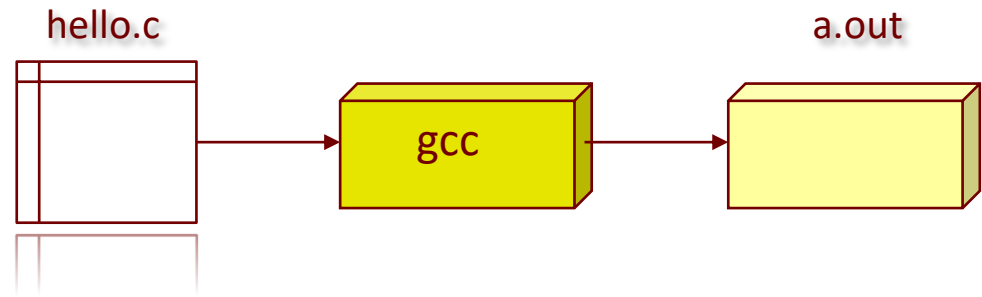
← Σχόλιο

← Οθόνη

← Τερματισμός προγράμματος

← Αλλαγή γραμμής

Μετάφραση του προγράμματος



❖ Στο τερματικό:

```
% ls
```

```
hello.c
```

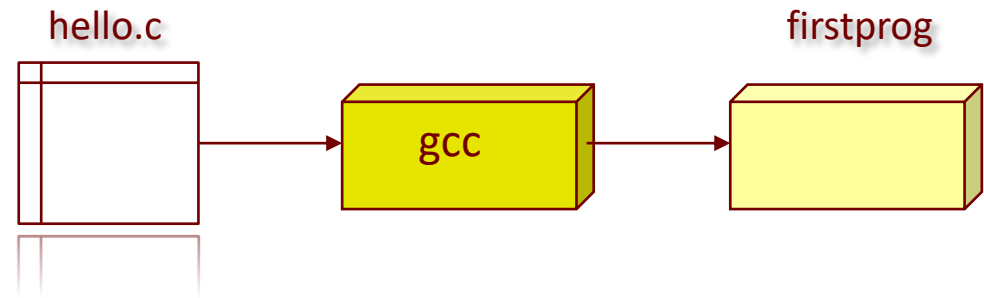
```
% gcc hello.c
```

```
% ls
```

```
a.out hello.c
```

❖ Ο μεταφραστής (`gcc`) ονομάζει το εκτελέσιμο “`a.out`”

Μετάφραση του προγράμματος με δικό μας όνομα



❖ Στο τερματικό:

```
% ls
```

```
hello.c
```

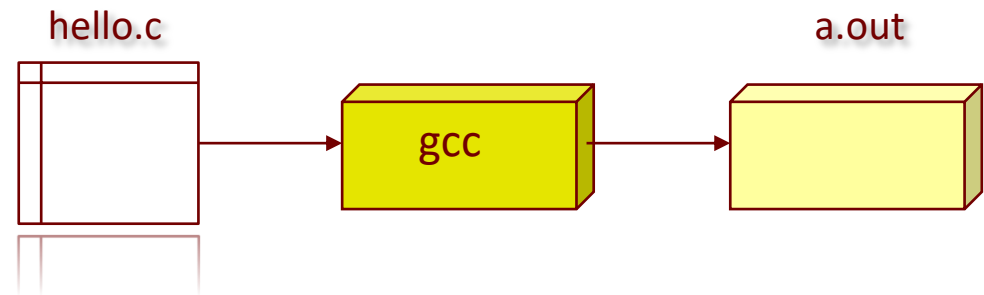
```
% gcc -o firstprog hello.c
```

```
% ls
```

```
firstprog hello.c
```

❖ Με το “-o” ο μεταφραστής αντί για “a.out” ονομάζει το εκτελέσιμο με ότι όνομα μας αρέσει.

Εκτέλεση του προγράμματος



❖ Στο τερματικό:

```
% ls
```

```
hello.c
```

```
% gcc hello.c
```

```
% ls
```

```
a.out hello.c
```

```
% ./a.out
```

```
Hello, World
```

```
%
```

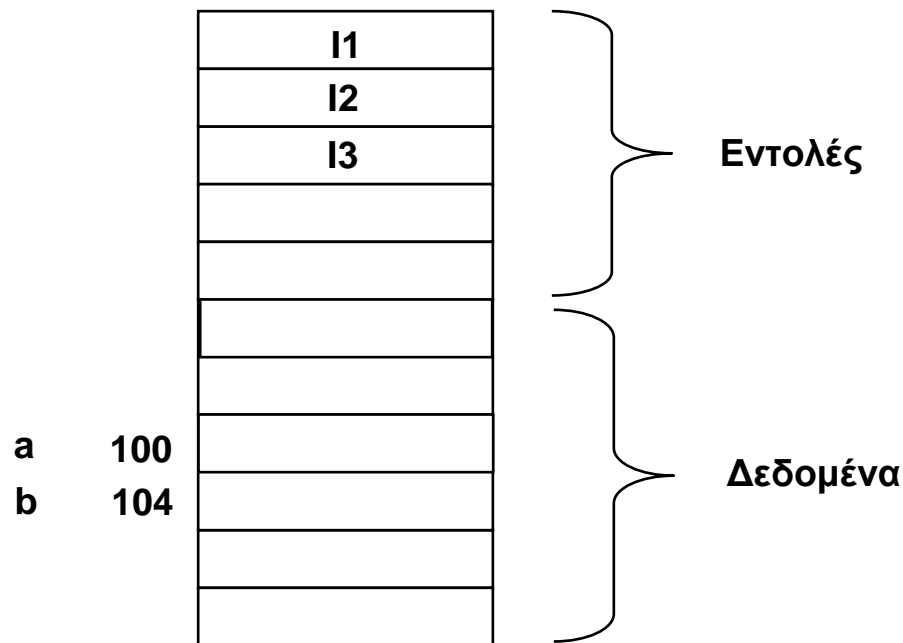
❖ Το “./” εννοεί το τρέχον directory – ίσως και να μην χρειάζεται.

- ❖ Πρόγραμμα = {δεδομένα} + {κώδικας (συναρτήσεις)}
- ❖ Συναρτήσεις = {main, ...}
- ❖ Δεδομένα = {μεταβλητές, σταθερές}
- ❖ Σταθερές = ποσότητες που δεν μεταβάλλονται κατά την εκτέλεση του προγράμματος
 - Π.χ. $\pi = 3.14$
- ❖ Μεταβλητές = ποσότητες που μεταβάλλονται
 - I/O, υπολογισμοί

- ❖ Τα δεδομένα αποθηκεύονται στη μνήμη του υπολογιστή
- ❖ Δηλώνοντας μια μεταβλητή δεσμεύω θέσεις στη μνήμη και καθορίζω ένα όνομα που χρησιμοποιώ για να αναφερθώ σε αυτή τη θέση μνήμης
- ❖ π.χ.
 - `int const n = 10;`
 - `int a;`

Εισαγωγικά

- ❖ Και οι εντολές αποθηκεύονται στη μνήμη του υπολογιστή
- ❖ Π.χ. εντολή I1: πρόσθεσε την τιμή της μεταβλητής a με αυτή της b → πρόσθεσε τα περιεχόμενα της θέσης 100 στα περιεχόμενα της θέσης 104



- ❖ Κάθε μεταβλητή, σταθερά έχει ένα τύπο
- ❖ Ο τύπος καθορίζει το μέγεθος του «κελιού» που θα δεσμευτεί στη μνήμη
 - char: 1 byte
 - int: 4 bytes (συνήθως)
 - float: 4 bytes
 - double: 8 bytes
- ❖ Προσδιοριστές
 - short int: 2 bytes
 - long int: 8 bytes
 - long double: ? bytes (≥ 10)

❖ Προσδιοριστές: unsigned

- Χωρίς πρόσημο (μόνο θετικοί) – όλα τα bits για την τιμή του αριθμού

❖ unsigned int, unsigned short int, unsigned char

- 32 bits, 16 bits, 8 bits
 - ✧ 0, 1, ..., 4294967295
 - ✧ 0, 1, ..., 65535
 - ✧ 0, 1, ..., 255

❖ int, short int, char

- 31 bits, 15bits, 7 bits (συν 1 για το πρόσημο)
 - ✧ -2147483648, ..., 2147483647
 - ✧ -32768, ..., 32767
 - ✧ -128, ..., 127

❖ Η βασικότερη και γενικότερη συνάρτηση εκτύπωσης είναι η “printf”.

- Αρχικά πρέπει να δώσω ως πρώτο όρισμα μια συμβολοσειρά με το TI ΤΥΠΟ θα εκτυπώσει
- Στη συνέχεια, τα επόμενα ορίσματα είναι οι αντίστοιχες μεταβλητές ή εκφράσεις

❖ Παράδειγμα:

```
int main() {  
    int x = 5;  
    float y = 1.2;  
  
    printf("%d", x);    /* %d ή %i για ακραίους */  
    printf("%f", y);    /* %f για πραγματικούς */  
    printf("x = %d and y = %f", x, y);  
    return 0;  
}
```

- ❖ Όσες μεταβλητές βρίσκονται εντός ενός μπλοκ εντολών (π.χ. μέσα σε μία συνάρτηση) είναι **τοπικές (local)** και μπορούν να χρησιμοποιηθούν μόνο εντός του μπλοκ.
- ❖ Όσες είναι εκτός των συναρτήσεων είναι **καθολικές (global)** και μπορούν να χρησιμοποιηθούν παντού.
- ❖ Περισσότερα αργότερα...

- ❖ Όλα τα δεδομένα σε ένα υπολογιστή κωδικοποιούνται ως ακολουθίες 0, 1
- ❖ Ένας χαρακτήρας κωδικοποιείται ως ακολουθία 0, 1
- ❖ Άρα στην ουσία ο υπολογιστής τον αντιλαμβάνεται σαν ένα αριθμό
 - Ο χαρακτήρας '0' αντιστοιχεί στον αριθμό 48
 - `char ch = 'x';`
 - Λέγοντας `ch = 'x'` είναι σαν να λέμε: βάλε στη μεταβλητή `ch` την τιμή (αριθμό) που αντιστοιχεί στο χαρακτήρα 'x'
 - `ch++;`
- ❖ Αριθμητική τιμή = κωδικός ASCII

Παράδειγμα με printf

```
#include <stdio.h>

int main()
{
    char ch = 'x';    /* Τοπική μεταβλητή τύπου χαρακτήρα */

    printf("ch = %d, ch = %c\n", ch, ch);

    return 0;
}
```

Ακέραιοι σε διάφορες μορφές...

```
int main() {  
    int x = 95;    /* 000...0 0101 1111 */  
  
    printf("%d", x);    /* 95 */  
    printf("%x", x);    /* 5f */  
    printf("%o", x);    /* 137 */  
    printf("%c", x);    /* _ */  
  
    x = 95;        /* Καταχωρώντας το 95 σε διάφορες μορφές */  
    x = 0x5F;  
    x = 0137;  
  
    x = 'd';        /* Ποιος αριθμός είναι αυτός; */  
    printf("%d", x);    /* 100 */  
    printf("%x", x);    /* 64 */  
    printf("%o", x);    /* 144 */  
    printf("%c", x);    /* d */  
    return 0;  
}
```

❖ Αριθμητικοί τελεστές

$+$, $-$, $*$, $/$, $\%$ (το τελευταίο μόνο για ακεραίους)

❖ Συγκριτικοί τελεστές

$>$, $<$, \leq , \geq , $==$, $!=$

❖ Λογικοί τελεστές

$\&\&$, $||$, $!$

❖ Υπάρχουν και κάποιοι άλλοι τελεστές που θα μας απασχολήσουν αργότερα

➤ Bitwise operators: \sim , $\&$, $|$, \wedge , \gg , \ll

- ❖ Αριθμητικοί τελεστές για δεδομένα ίδιου τύπου κυρίως (π.χ. πρόσθεση δύο ακεραίων)
- ❖ Όμως, μπορούμε να κάνουμε και πράξεις με μεταβλητές διαφορετικού τύπου, π.χ.

```
int x; float f;  
f = f+x;
```

 - Γίνεται εσωτερική μετατροπή των «κατώτερων» τύπων σε «ανώτερους»
 - Και το αποτέλεσμα ανώτερου τύπου
- ❖ Τέτοιες μετατροπές γίνονται αυτόματα αλλά μπορούμε να τις ζητήσουμε και εμείς σε ένα πρόγραμμα με το μηχανισμό των “type casts”
 - $f + x$ (το x μετατρέπεται αυτόματα σε float)
 - $f + ((\text{float}) x)$ (cast του προγραμματιστή)

❖ Τελεστές σύντμησης:

`++, --, +=, -=, *=, /=`

❖ Παράδειγμα

- `++i` και `i++` (pre-increment, post-increment)
- `i=i+1` και `i+=1`

❖ Παράδειγμα

```
i = 3;
```

```
x = ++i; /* πρώτα γίνεται η αύξηση και μετά η αποτίμηση της έκφρασης */
```

```
x = i++; /* πρώτα γίνεται η αποτίμηση της έκφρασης και μετά η αύξηση */
```

```
i = i++ + ++i; /* εδώ τι τιμή θα πάρει τελικά το i? */
```

μεταβλητή = συνθήκη ? τιμή1 : τιμή2;

❖ Η εκτέλεση ισοδυναμεί με:

```
if (συνθήκη)
    μεταβλητή = τιμή1;
else
    μεταβλητή = τιμή2;
```

❖ Παράδειγμα:

```
x = (y > 0) ? 1 : 0;
```

2 «στυλ» σταθερών

❖ Στη java το “final” μπορεί να χρησιμοποιηθεί για να ορίσει «σταθερές»

❖ Στη C υπάρχουν 2 τρόποι να οριστούν σταθερές:

1. Με προσθήκη του «const» στον τύπο της δήλωσης, π.χ.

```
const int x = 5;    /* Δεν μπορεί να αλλάξει τιμή */
```

2. Με ορισμό σταθεράς προεπεξεργαστή (#define)

```
#define M 10          /* Το σύνηθες ... */
```

```
#define PI 3.14
```

```
#define NEWLINE '\n'
```

❖ Διαφορά:

- Οι const καταλαμβάνουν μνήμη για αποθήκευση
- Οι #define ΑΝΤΙΚΑΘΙΣΤΑΝΤΑΙ ΠΡΙΝ ΓΙΝΕΙ Η ΜΕΤΑΦΡΑΣΗ του προγράμματος (και άρα δεν υπάρχουν στο εκτελέσιμο)

Διάβασμα (scanf) / εκτύπωση (printf)

```
#include <stdio.h> /* Απαραίτητο */

char c;                /* Καθολική μεταβλητή */

int main() {           /* Συνάρτηση main */
    int i;              /* Τοπικές δηλώσεις ΠΑΝΤΑ στην αρχή της συνάρτησης */
    float f;

    printf("Dwse 1 xaraktira, 1 akeraio kai enan pragmatiko\n");
    scanf("%c", &c);
    scanf("%d%f", &i, &f);      /* Η scanf ΘΕΛΕΙ & στις μεταβλητές */
    printf("c = %c, i = %d, f = %f\n", c, i, f);
    return 0;
}
```

Εισαγωγή στη C

Εντολές



ΜΥΥ 502

❖ Όπως και στην java:

- if
- if-else
- switch
- for
- while
- do-while
- break

❖ Επιπλέον:

- goto

if & if-else

```
if (condition) {  
    statements;  
}
```

```
if (condition) {  
    statements;  
} else {  
    statements;  
}
```

Αν υπάρχει ΜΟΝΟ ΈΝΑ statement, όπως και στην Java, δε χρειάζονται οι αγκύλες. Π.χ. ο παρακάτω κώδικας

```
if (x == 3)  
    x++;  
    y++;  
z = x+y;
```

είναι ισοδύναμος με:

```
if (x == 3) {  
    x++;  
}  
y++;  
z = x+y;
```

```
switch (variable) {  
    case const1:  
        statements;  
        break;  
    case const2:  
        statements;  
        break;  
    ...  
    default:  
        statements;  
        break;  
}
```



```
switch (variable) {  
    case const1:  
        statements;  
        break;  
    case const2:  
        statements;  
        break;  
    ...  
    default:  
        statements;  
        break;  
}
```

```
switch ( choice )  
{  
    case 'a':  
    case 'A':  
        do_thing_1();  
        break;  
  
    case 'b':  
    case 'B':  
        do_thing_2();  
        break;  
    ...  
  
    default:  
        printf("Wrong choice.");  
}
```

❖ Προσοχή:

- Αν δεν υπάρχει break, η εκτέλεση ενός case συνεχίζεται με τον κώδικα του επόμενου case...

while & do-while

```
while (condition) {  
    statements;  
}
```

```
x = 0;  
while (x < 10)  
    x++;  
  
x = 0;  
while (x < 10);  
    x++;
```

```
do {  
    statements;  
} while (condition);
```

Αν υπάρχει ΜΟΝΟ ΈΝΑ statement, όπως και στη Java, δε χρειάζονται οι αγκύλες.

for (I)

```
for (initialization; condition; iteration) {  
    statements;  
}
```

```
/* Ισοδύναμος κώδικας: */  
initialization;  
while (condition) {  
    statements;  
    iteration;  
}
```

Αν υπάρχει ΜΟΝΟ ΈΝΑ statement, όπως και στην Java, δε χρειάζονται οι αγκύλες.

- ❖ Τα initialization / iteration μπορούν να περιέχουν πολλές εκφράσεις, χωρισμένες με κόμμα.

for (II)

- ❖ Τα initialization / iteration μπορούν να περιέχουν πολλές εκφράσεις, χωρισμένες με κόμμα.

Π.χ.

```
int i, sum;  
sum = 0;  
for (i = 1; i <= 10; i++) {  
    sum += i;  
}
```

```
for (i = 1, sum = 0; i <= 10; sum += (i++))  
    ;
```

❖ Δεν υπάρχει το for each

~~for (δήλωση : συλλογή)~~

της Java.

- ❖ Έξοδος από switch ή από βρόχους for/while/do, π.χ.

```
while (1) {  
    if (w == 3) {  
        break;    /* Βγαίνει αμέσως μετά το while */  
    }  
    ...  
}
```

- ❖ Δεν υπάρχει ονοματισμένο break (τύπου break <label>)

- ❖ Η εκτέλεση μεταπηδά σε συγκεκριμένη ετικέτα, π.χ.

```
    if (x == 1) {  
        goto before;  
    }  
    y = 2;  
    goto after;  
before: y = 1;  
after:  ...
```

- ❖ Επικίνδυνη / μη-προβλέψιμη εντολή, π.χ.

```
    while (1) {  
        if (w == 3) {  
            Strange: x=1; ...  
        }  
    }  
    ...  
    if (condition) goto Strange;
```

- ❖ «Ακατάλληλη» δια ανηλίκους. Κακή προγραμματιστική τεχνική. Δεν πρέπει να χρησιμοποιείται σχεδόν ποτέ.

Αγκύλες και μπλοκ κώδικα

- ❖ Γράψτε τον παρακάτω κώδικα ΧΩΡΙΣ αγκύλες, όπου γίνεται:

```
if (i >= 81) {  
    x = 3;  
}  
if (row >= 1 && col<= 9) {  
    q = 5;  
    y = x;  
}  
else {  
    for (S[row][col] = 1; S[row][col] <= 9; S[row][col]++) {  
        if ( sudoku_solve(i) ) {  
            return (1);  
        }  
    }  
}
```


❖ Πρώτη προσπάθεια:

```
if (i >= 81) {  
    x = 3;  
}  
if (row >= 1 && col<= 9) {  
    q = 5;  
    y = x;  
}  
else {  
    for (S[row][col] = 1; S[row][col] <= 9; S[row][col]++) {  
        if ( sudoku_solve(i) ) {  
            return (1);  
        }  
    }  
}
```

❖ Δεύτερη προσπάθεια:

```
if (i >= 81
    x = 3;

if (row >= 1 && col<= 9) {
    q = 5;
    y = x;
}
else {
    for (S[row][col] = 1; S[row][col] <= 9; S[row][col]++) {
        if ( sudoku_solve(i) )
            return (1);
    }
}
```

❖ Τρίτη προσπάθεια:

```
if (i >= 81
    x = 3;

if (row >= 1 && col<= 9) {
    q = 5;
    y = x;
}
else {
    for (S[row][col] = 1; S[row][col] <= 9; S[row][col]++)
        if ( sudoku_solve(i) )
            return (1);

}
```

❖ Τελικός κώδικας:

```
if (i >= 81
    x = 3;

if (row >= 1 && col<= 9) {
    q = 5;
    y = x;
}
else
    for (S[row][col] = 1; S[row][col] <= 9; S[row][col]++)
        if ( sudoku_solve(i) )
            return (1);
```

❖ ΠΑΝΤΑ ΝΑ ΒΑΖΕΤΕ ΑΓΚΥΛΕΣ, ΑΚΟΜΑ ΚΑΙ ΉΤΑΝ ΔΕΝ ΧΡΕΙΑΖΕΤΑΙ