ΜΥΥ502

Προγραμματισμός Συστημάτων

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ ΠΑΝΕΠΙΣΤΗΜΙΟΥ ΙΩΑΝΝΙΝΩΝ

Β. Δημακόπουλος

dimako@cse.uoi.gr

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

Προγραμματισμός Συστημάτων - Αντικείμενο

- 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)

Προγραμματισμός Συστημάτων - Ύλη

- 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). "

Προγραμματισμός Συστημάτων – Διάρθρωση θεμάτων

- 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

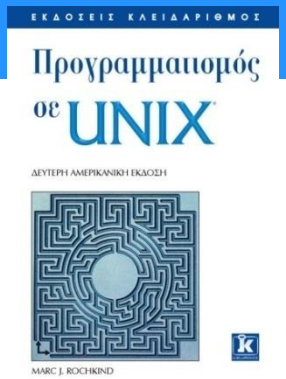
  - Προγραμματισμός συστημάτων POSIX και προχωρημένα θέματα
    - Εμβάθυνση σε προχωρημένες δυνατότητες της C
    - Γνωριμία με διαδικασίες και εργαλεία ανάπτυξης εφαρμογών συστήματος
    - Βασικές κλήσεις POSIX (διεργασίες, σήματα, επικοινωνίες, νήματα κλπ)
    - Άλλα προχωρημένα θέματα και τεχνικές

    Π2: Περίπου 50% της ύλης

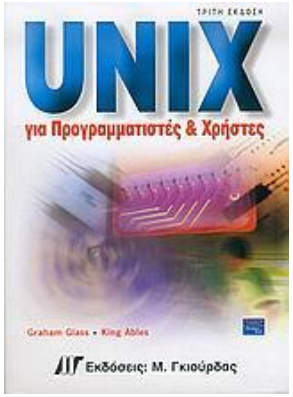
Συγγράμματα

- 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

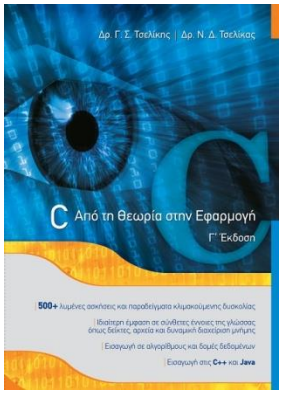
Εύδοξος



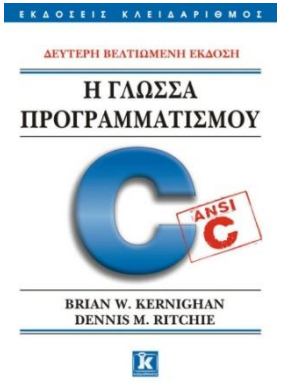
- ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΣΕ UNIX,
  M.J. Rochkind (2007)
  - Εξαιρετικό βιβλίο για προγραμματισμό συστημάτων UNIX
  - Υποθέτει γνώση της C



- 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 ΕΦΑΡΜΟΓΗ,
  Γ. Τσελίκης, Ν. Τσελίκας (νέα έκδοση, 2023)
  - 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.

Ιστοσελίδες

- 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

Ώρες μαθήματος

- Διαλέξεις:
  - Δευτέρα: 12:00 – 14:00
  - Τετάρτη: 12:00 – 14:00 (*)
  - Αίθουσα: Ι5
- Εργαστήρια (το πιθανότερο):
  - Τρίτη: 14:00 – 18:00
  - ΠΕΠ Ι, ΠΕΠ ΙΙ, ΠΕΛΣ
- Ώρες γραφείου διδάσκοντα:
  - Τρίτη: 09:00 – 11:00
  - Β33

	Δε	Τρ	Τε	Πε	Πα
08:00					
09:00		Γρφ			
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/

Εργαστήρια (αν εγγράφεστε **ΠΡΩΤΗ ΦΟΡΑ ΣΤΟ ΕΡΓΑΣΤΗΡΙΟ**)

- Σκοπός/λειτουργία εργαστηρίων:
  - **«Φροντιστηριακού» τύπου για ενίσχυση διδασκαλίας**
  - Σχεδιασμός προγράμματος και υλοποίηση «επί τόπου»
  - (Πολύ) έμπειροι μεταπτυχιακοί και προσωπικό για να σας βοηθήσουν

- Οργανωτικά:
  - **Θα υπάρξει φόρμα εγγραφής τις επόμενες ημέρες**
  - **Ξεκινούν (λογικά) σε 2 εβδομάδες – θα σας ενημερώσω εγκαίρως**
  - 14:00 – 18:00 (2 βάρδιες λογικά, αν χρειαστεί θα μπει και τρίτη βάρδια)
  - **Παρουσίες**
    - Οι κανόνες λειτουργίας είναι αυστηροί, δεν πρέπει να καθυστερείτε

Όσοι ΠΕΡΑΣΟΥΝ το εργαστήριο, το κατοχυρώνουν για πάντα

Και έρχονται μόνο στις τελικές εξετάσεις.

- 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…

Τι σημαίνει «**ΠΕΡΝΑΩ**» το εργαστήριο

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

FAQ

- ΕΡΩΤΗΣΗ: *Αν κάνω όλες τις παρουσίες περνάω;*
- ΑΠΑΝΤΗΣΗ: ΟΧΙ, ο βαθμός έχει μέσα και τις προόδους (βλ. παραπάνω)

- 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

Εργαστήρια (για όσους είχαν **ΕΓΓΡΑΦΕΙ ΠΑΛΑΙΑ** και απέτυχαν)

- Αφορά μόνο όσους:
  1. Έχουν ξαναεγγραφεί στο εργαστήριο κατά το παρελθόν ΚΑΙ
  2. Δεν έχουν περάσει το εργαστήριο ποτέ

- **Εγγραφή στα εργαστήρια:**
  - **Θα υπάρξει φόρμα εγγραφής πριν την 1η πρόοδο**
  - **Υποχρεωτική**, πλήρης (ηλεκτρονική) εγγραφή
  - **Αλλιώς ΔΕΝ ΘΑ ΜΠΟΡΕΣΟΥΝ ΝΑ ΕΞΕΤΑΣΤΟΥΝ**

- **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 **≥ 4,5**.

FAQ

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

- 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: Έως αδύνατο λόγω χώρου. Αν όμως θέλετε να ασχοληθείτε μόνοι σας θα υπάρχουν και οι ασκήσεις και οι απαντήσεις και οι βοηθοί.

- ΕΡΩΤΗΣΗ: *Μπορώ να κρατήσω τον παλιό (αποτυχημένο) βαθμό και να έρθω κατευθείαν στις εξετάσεις όπου θα γράψω πολύ καλά;*
- ΑΠΑΝΤΗΣΗ: **ΟΧΙ. Όσοι είχαν αποτύχει, ΠΡΕΠΕΙ ΝΑ ΞΑΝΑΔΩΣΟΥΝ ΤΙΣ ΠΡΟΟΔΟΥΣ**

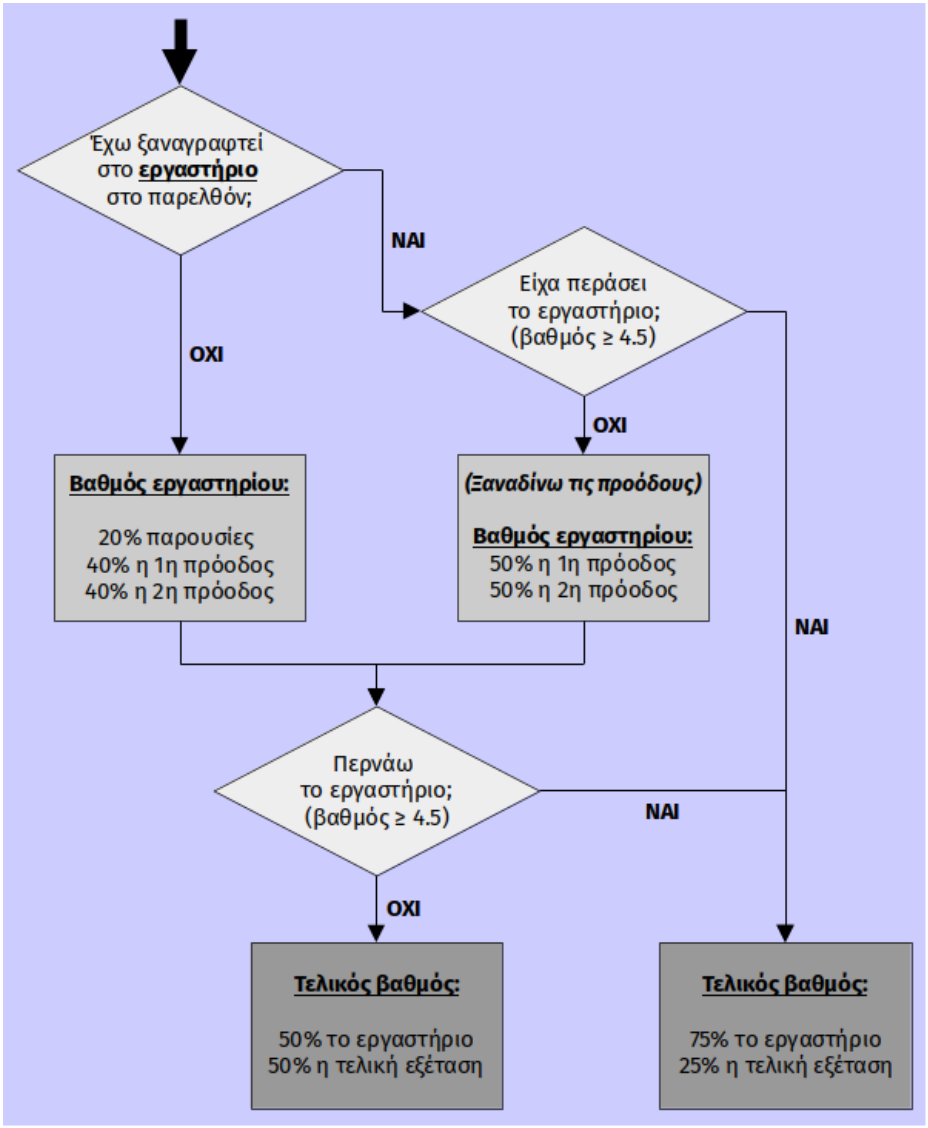
Τελική βαθμολόγηση μαθήματος

- Για όσους περάσουν το εργαστήριο:
  - **75% εργαστήριο + 25% τελικές εξετάσεις**

- Για όσους **δεν** περάσουν το εργαστήριο:
  - **50% εργαστήριο + 50% τελικές εξετάσεις**
    (άρα θα έχουν ευκαιρία να περάσουν το μάθημα με καλό βαθμό στις εξετάσεις)

Προφανώς, ο τελικός βαθμός θα πρέπει να είναι >= 4.5 για επιτυχία.

Για όσους θέλουν αλγόριθμο…



Το σημερινό μάθημα

Εισαγωγικά στοιχεία για τη C

Η γλώσσα 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

- IEEE, Αυγ. 2022
  https://spectrum.ieee.org/top-programming-languages/

Δημοτικότητα της C (TIOBE index, Σεπτ. 2021)

- TIOBE index: βάσει hits σε 25 top search engines

Sep 2021	Sep 2020	Change	Programming Language	Ratings	Change
1	1		C	11.83%	-4.12%
2	3	▲	Python	11.67%	+1.20%
3	2	▼	Java	11.12%	-2.37%
4	4		C++	7.13%	+0.01%
5	5		C#	5.78%	+1.20%
6	6		Visual Basic	4.62%	+0.50%
7	7		JavaScript	2.55%	+0.01%

Δημοτικότητα της C (TIOBE index, Long-term)

- TIOBE index: βάσει hits σε 25 top search engines
- **Όλες οι κορυφαίες με βάση την C, σε όλες τις κατατάξεις!**

Programming Language	2021	2016	2011	2006	2001	1996	1991	1986
C	1	2	2	2	1	1	1	1
Java	2	1	1	1	3	15	-	-
Python	3	5	6	8	25	24	-	-
C++	4	3	3	3	2	2	2	6
C#	5	4	5	7	13	-	-	-
Visual Basic	6	13	-	-	-	-	-	-
JavaScript	7	7	10	9	9	20	-	-
PHP	8	6	4	4	10	-	-	-

Ενέργεια, ταχύτητα, κατανάλωση μνήμης

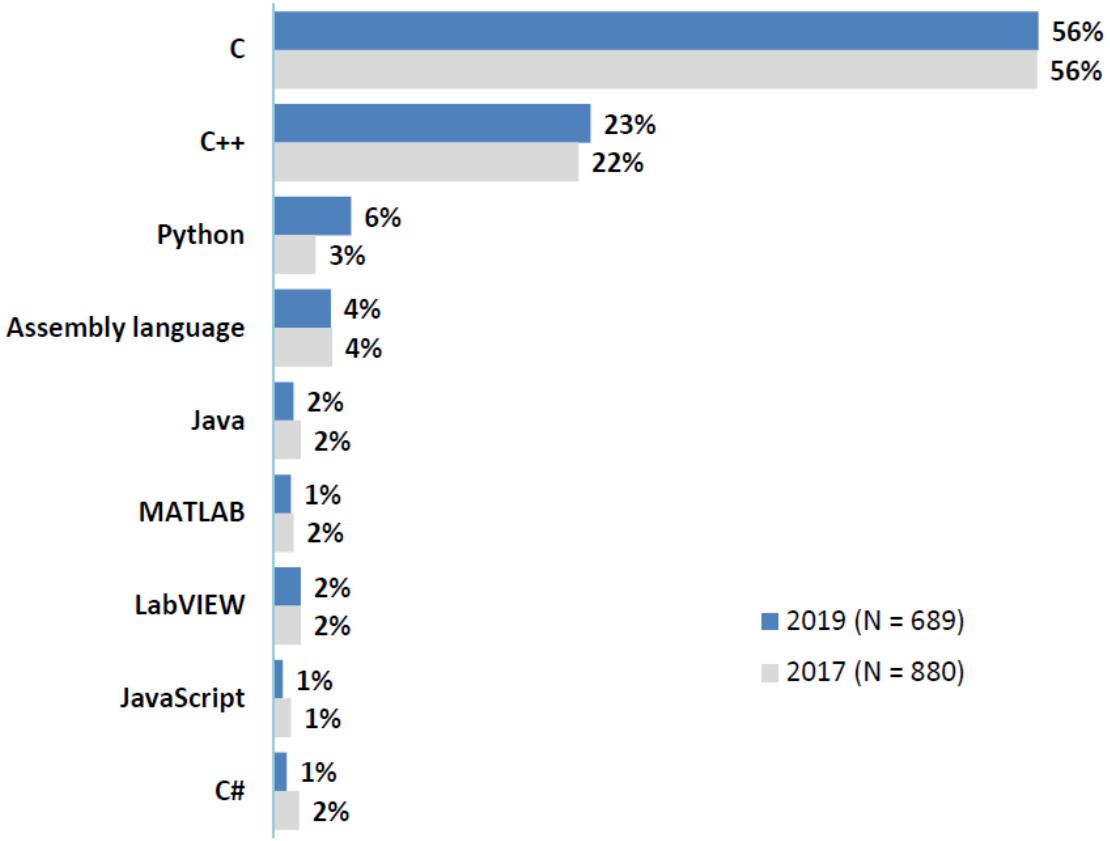
Table 4
Normalized global results for Energy, Time, and Memory.

Total					
	Energy (J)		Time (ms)		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

"Ranking programming languages by energy efficiency", *Science of Computer Programming*, 2021

Στον χώρο των ενσωματωμένων συστημάτων

My *current* embedded project is programmed mostly in:



EETimes embedded 2019 Embedded Markets Study

Κι άλλα…

- Το UNIX γράφτηκε σε C
- Ο πυρήνας του Linux είναι γραμμένος σε C
- Σχεδόν όλες οι εφαρμογές συστήματος είναι σε C
- Άπειρες εφαρμογές 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 έχει κάποιες ευκολίες που έχουν άλλες γλώσσες και δεν βολεύει πάντα για εφαρμογές χρήστη, ειδικά όταν υπάρχει γραφική αλληλεπίδραση

Αλήθειες…

"The 'C' students run the world."

— Harry S. Truman

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;
}
```

Η C δεν έχει...

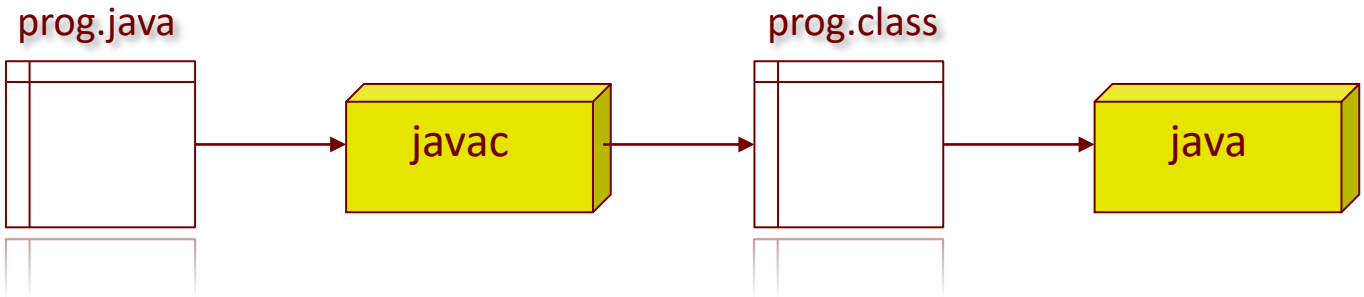
- Κλάσεις
  - Μόνο δεδομένα (μεταβλητές) και συναρτήσεις
  - Η συνάρτηση main() είναι αυτή που εκτελείται αρχικά
- Boolean
  - Με ακεραίους «προσομοιώνουμε» τα boolean
  - Το **0** θεωρείται FALSE
  - Οτιδήποτε μη-μηδενικό θεωρείται TRUE
- Strings (τουλάχιστον όπως τα χειρίζεται η Java)
  - Χειρισμός μέσω πινάκων και δεικτών
- try … catch μπλοκ (exceptions)
  - Δεν υπάρχει ανάλογο, μόνο μέσω συναρτήσεων συστήματος

Η C έχει...

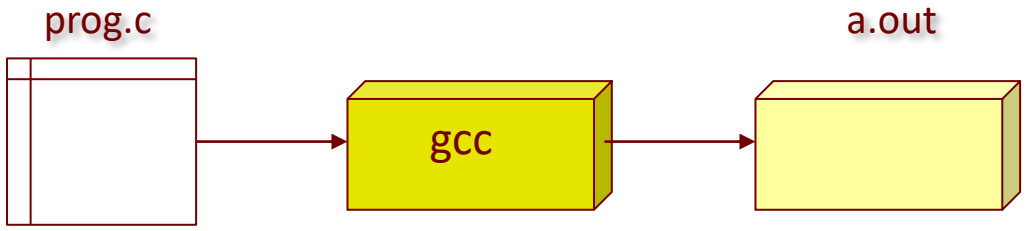
- Pointers (δείκτες)!
  - Όχι μόνο απλό πέρασμα με αναφορά
- «Ελευθερία» στους τύπους των δεδομένων (π.χ. int/short/char είναι πάνω-κάτω ίδιοι) και δεν γίνεται πλήρης έλεγχος κατά τη χρήση τους.
- «Ελευθερία» στη διαχείριση της μνήμης
  - Επαφίεται πλήρως στον προγραμματιστή
  - Η java έχει garbage collector που αυτόματα αποδεσμεύει άχρηστη μνήμη

Java vs C

- Η Java είναι (βασικά) *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 (βασικά) *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

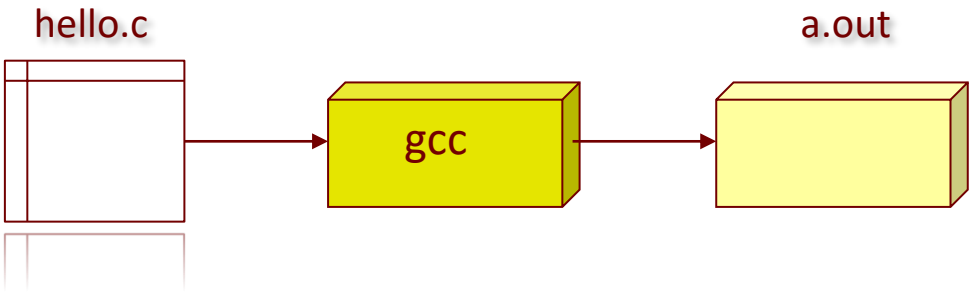
Σχόλιο

Οθόνη

Τερματισμός προγράμματος

Αλλαγή γραμμής

Μετάφραση του προγράμματος

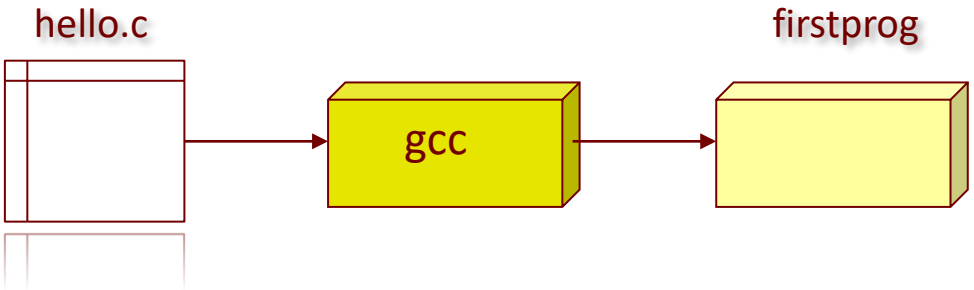


- Στο τερματικό:

```
% 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

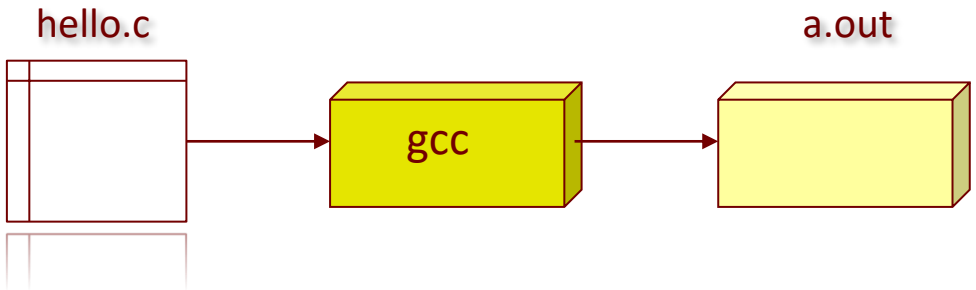


- Στο τερματικό:

```
% 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.

Εισαγωγικά

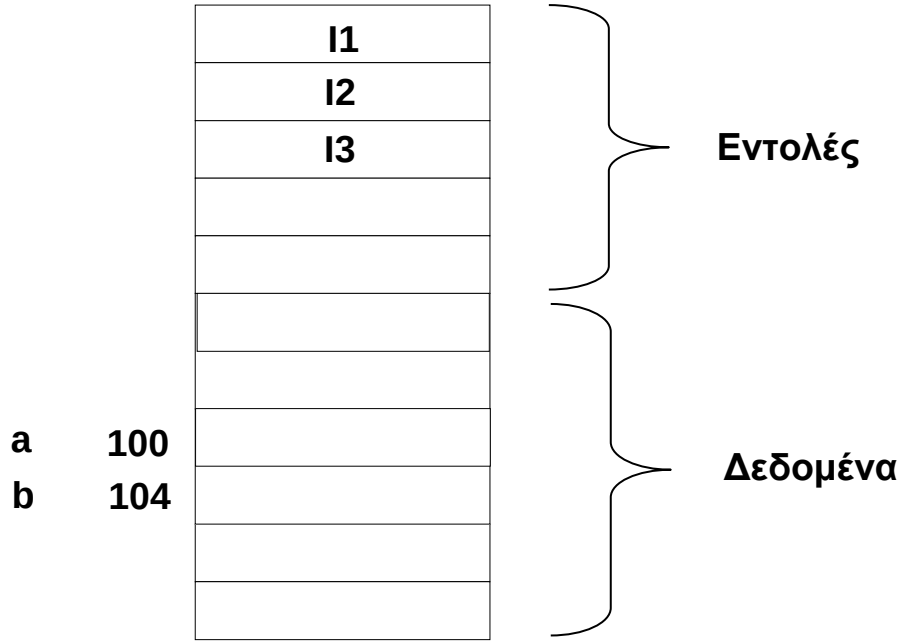
- Πρόγραμμα = {δεδομένα} + {κώδικας (συναρτήσεις)}
- Συναρτήσεις = {main, …}
- Δεδομένα = {μεταβλητές, σταθερές}
- Σταθερές = ποσότητες που δεν μεταβάλλονται κατά την εκτέλεση του προγράμματος
  - Π.χ. Pi = 3.14
- Μεταβλητές = ποσότητες που μεταβάλλονται
  - I/O, υπολογισμοί

Εισαγωγικά

- 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



Μεταβλητές & τύποι δεδομένων

- 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)

Μεταβλητές & τύποι δεδομένων

- 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`

Εκτυπώνοντας τις τιμές των μεταβλητών

- Η βασικότερη και γενικότερη συνάρτηση εκτύπωσης είναι η “printf”.
  - Αρχικά πρέπει να δώσω ως πρώτο όρισμα μια συμβολοσειρά με το ΤΙ 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 ακεραίους */
   printf("%f", y);    /* %f για πραγματικούς */
   printf("x = %d and y = %f", x, y);
   return 0;
}
```

Καθολικές / τοπικές μεταβλητές

- 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…

Χαρακτήρες

- 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;
}
```

Τελεστές

- Αριθμητικοί τελεστές

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

- Συγκριτικοί τελεστές

  `>, <, <=, >=, ==, !=`

- Λογικοί τελεστές

  `&&, ||, !`

- Υπάρχουν και κάποιοι άλλοι τελεστές που θα μας απασχολήσουν αργότερα
  - Bitwise operators: `~, &, |, ^, >>, <<`

Πράξεις και μετατροπές

- 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 + ((float) x)` (cast του προγραμματιστή)

Τελεστές

- 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? */
```

Ο «τριαδικός» τελεστής

```
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

Εντολές ελέγχου ροής

- 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

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

Εντολές ελέγχου – switch

```
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++))
   ;
```

for (III)

- Δεν υπάρχει το for each

  ~~for (δήλωση : συλλογή)~~

  της Java.

break

- 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>)

goto

- 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);
    }
  }
}
```

Αγκύλες και μπλοκ κώδικα

- 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);
    }
  }
}
```

Αγκύλες και μπλοκ κώδικα

- 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);

  }
}
```

Αγκύλες και μπλοκ κώδικα

- 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);


}
```

Αγκύλες και μπλοκ κώδικα

- 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 ΑΝ ΔΕΝ ΧΡΕΙΑΖΕΤΑΙ