

Διάλεξη 4 - Live Coding, Git και Τελεστές

Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Εισαγωγή στον Προγραμματισμό

Θανάσης Αυγερινός

Ανακοινώσεις / Διευκρινίσεις

- Η εργασία 0 θα βγει σήμερα! Προθεσμία στις 23:59, 6 Νοεμβρίου 2024

Την Προηγούμενη Φορά

- Τύπωμα μεταβλητών continued
- Συνέχεια του Hello World που αφήσαμε την άλλη φορά
- Συναρτήσεις και Ακέραιοι (Μαθηματικά vs C)

Σήμερα

- Pair Programming
- Git
- Τελεστές

GitHub

Pair Programming

Pair programming is a [software development](#) technique in which two [programmers](#) work together at one workstation. One, the *driver*, writes [code](#) while the other, the *observer* or *navigator*, [reviews](#) each line of code as it is typed in. The two programmers switch roles frequently.



Χρειάζομαι 2-3 εθελοντές/ντριες

ssh ubuntu@54.152.32.85

password: ilovedi@uoa

Υπολογισμός Βαθμολογίας Πρωτοετών

- $50\% * \text{Τελική Εξέταση} + 30\% * \text{Ασκήσεις} + 20\% * \text{Εργαστήριο}$
- Έστω ότι χρησιμοποιούμε 3 ακεραίους για να αναπαραστήσουμε τα αποτελέσματα [0 - 100].
 - `int final_exam`
 - `int homework`
 - `int lab`
- Στα μαθηματικά:
 - $\text{grade}(\text{final_exam}, \text{homework}, \text{lab}) = \text{final_exam} \times 50\% + \text{homework} \times 30\% + \text{lab} \times 20\%$
- Έλεγχος ορθότητας: `grade(70, 80, 100) == 79?`
- Σε C;

Πρόγραμμα Υπολογισμού Βαθμολογίας 1/2

```
#include <stdio.h>

#include <stdlib.h>

// Compute grades using the class formula

int grade(int final_exam, int homework, int lab) {

    return final_exam * 50 / 100 + homework * 30 / 100 + lab * 20 / 100;

}

int main(int argc, char **argv) {

    if (argc != 4) {

        printf("Program needs to be called as `./prog final_exam homework lab`\n");

        return 1;

    }

    int final_exam = atoi(argv[1]);

    int homework = atoi(argv[2]);

    int lab = atoi(argv[3]);

    printf("Grade: %d\n", grade(final_exam, homework, lab));

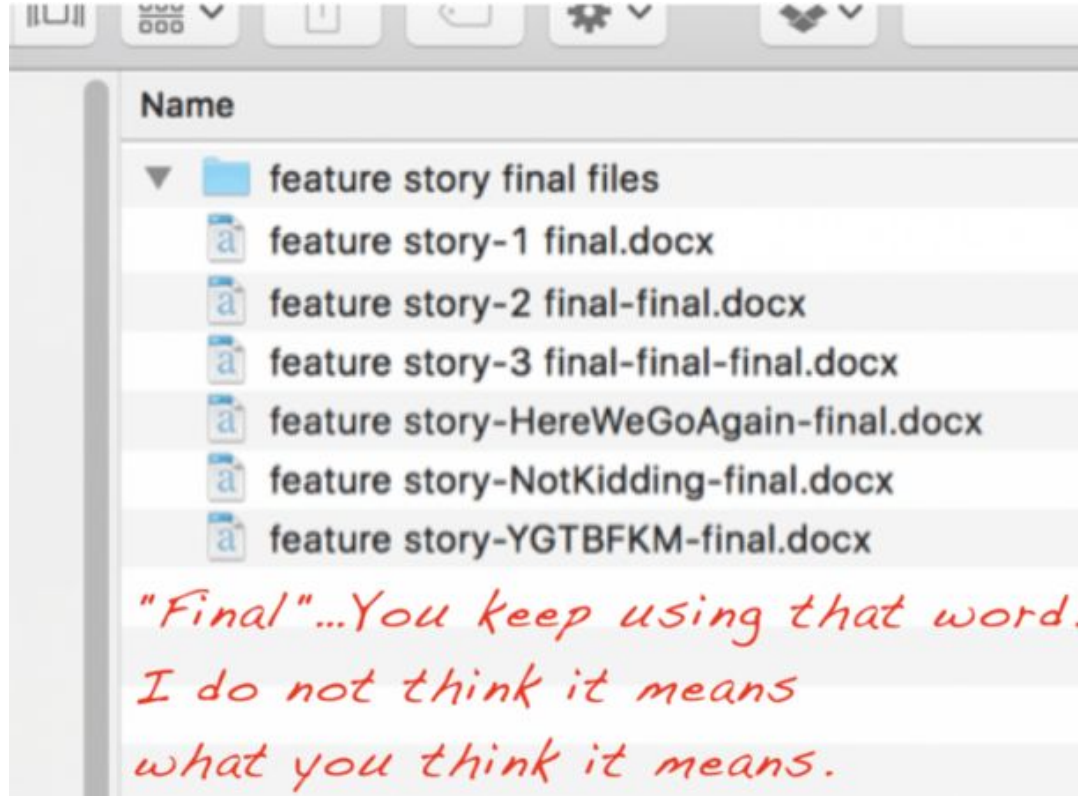
    return 0;

}
```


Πρόγραμμα Υπολογισμού Βαθμολογίας 1/2

- Η συνάρτηση `main` έχει μεταβλητό αριθμό ορισμάτων. Η παράμετρος `argc` λέει πόσα ορίσματα περάσαμε στο πρόγραμμα, ενώ η παράμετρος `argv` περιέχει το κείμενο που αντιστοιχεί σε κάθε όρισμα.
 - `argv[1]` περιέχει το 1ο όρισμα του προγράμματος, `argv[2]` το 2ο, κ.ο.κ.
- Η συνάρτηση βιβλιοθήκης (`stdlib.h`) `atoi` μετατρέπει ένα όρισμα από αλφαριθμητικό (ASCII) σε έναν ακέραιο (`integer`).
- Εάν σας ενοχλεί το πόσο "μαγική" φαίνεται η `main` και τα ορίσματά της, μην εξάπτεστε! Είναι μια από τις δυσκολότερες συναρτήσεις που θα συναντήσουμε στην C και τις επόμενες εβδομάδες θα χτίσουμε το υπόβαθρο για να γίνει πιο κατανοητή.

Version Control & Git



Version Control

- Ένα σύστημα που επιτρέπει να παρακολουθούμε όλες τις αλλαγές που γίνονται στον κώδικα και να κρατάμε ιστορικό αρχείο.

Γιατί να θέλουμε κάτι τέτοιο;;

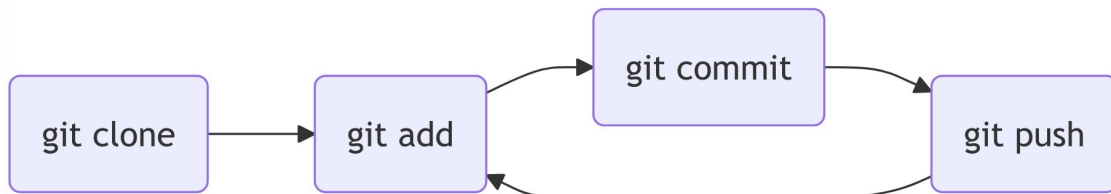
- Τα περισσότερα προγράμματα αλλάζουν με τον χρόνο - δεν υπάρχει "τελική μορφή"
- Κάνουμε αλλαγές και κρατάμε μόνο τα αρχεία που χρειαζόμαστε χωρίς να χάνουμε πληροφορία
- Μπορούμε να ανατρέξουμε σε οποιαδήποτε αλλαγή έγινε ανά πάσα στιγμή

Git

- Είναι το δημοφιλέστερο πρόγραμμα Version Control σήμερα (~90% των χρηστών/project)
- Πρώτη έκδοση γράφτηκε από τον Linus Torvalds το 2005
- Μια δημοφιλής και δωρεάν (για την ώρα) πλατφόρμα στην οποία μπορούμε να αποθηκεύσουμε τον κώδικά μας είναι το <https://github.com/>
- Θα το χρησιμοποιήσουμε για την γραφή και υποβολή των εργασιών μας

Git Development Cycle

Σε επίπεδο βασικής χρήσης (αυτό που χρειαζόμαστε για το μάθημα), η γραφή Κώδικα και η αποθήκευση με git περιλαμβάνει 4 βήματα:



THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.



Clone Repository

`git clone`: Αντιγράφει το repository (αποθετήριο) με όλα τα αρχεία σε έναν τοπικό φάκελο.

```
git clone git@github.com:progintro/hw0-barbouni-2005.git
```

- Αν ελέγξουμε τον φάκελο παρατηρούμε ότι έχει ένα README.md αρχείο.
- Τρέχοντας `git status` μέσα στον φάκελο μπορούμε να δούμε αν υπάρχουν καθόλου αλλαγές.
- Η παραπάνω εντολή θα δημιουργήσει έναν τοπικό φάκελο (αν δεν υπάρχει) με το όνομα `hw0-barbouni-2005`
- Τα βήματα που ακολουθούμε εδώ, προϋποθέτουν ότι έχετε φτιάξει ένα κλειδί για τον Github λογαριασμό σας και έχετε τελειώσει configuration όπως περιγράφεται στο Φυλλάδιο 1 του εργαστηρίου.

Προσθήκη Αρχείου

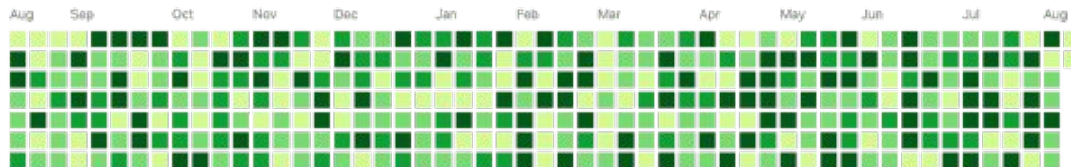
`git add`: Προσθέτει τα αρχεία στο staging area (προσωρινή λίστα) για μελλοντική αποθήκευση στο repository.

- Δημιούργησε ένα καινούριο αρχείο, π.χ., `touch command/solution.txt`.
- Τρέξε `git status` - θα παρατηρήσεις ότι το καινούριο αρχείο καταγράφεται ως `untracked`.
- Για να το προσθέσουμε: `git add command/solution.txt`
- Τρέξε `git status` ξανά - θα παρατηρήσεις ότι το αρχείο είναι έτοιμο να γίνει `commit` (καταχωρηθεί).

Καταχώρηση Αλλαγών

`git commit`: Αποθηκεύει τις αλλαγές στο τοπικό repository.

- Τρέξε `git commit` και δώσε ένα μήνυμα που περιγράφει την αλλαγή σου.
- Τρέξε `git status` και πάλι - τώρα το μόνο που μένει είναι να σώσουμε τις αλλαγές μας και εκτός του υπολογιστή μας.



**Looking for someone
who can commit**

Ανέβασμα Αλλαγών

`git push`: Ανεβάζει τις τοπικές αλλαγές στον απομακρυσμένο server (εξυπηρετητή).

- Έλεγξε ότι οι αλλαγές σου εμφανίζονται στο Github!

Κατέβασμα Αλλαγών

`git pull`: Κατεβάζει αλλαγές από τον απομακρυσμένο server (αν υπάρχουν).

- Χρήσιμο όταν έχεις πολλούς προγραμματιστές ή πολλούς υπολογιστές και δεν θέλεις να κάνεις clone κάθε φορά.
- Κάνε κάποιες αλλαγές στο repository από το GitHub UI και μετά τρέξε `git pull`.

Don't forget to add
your files before
committing &
pushing!



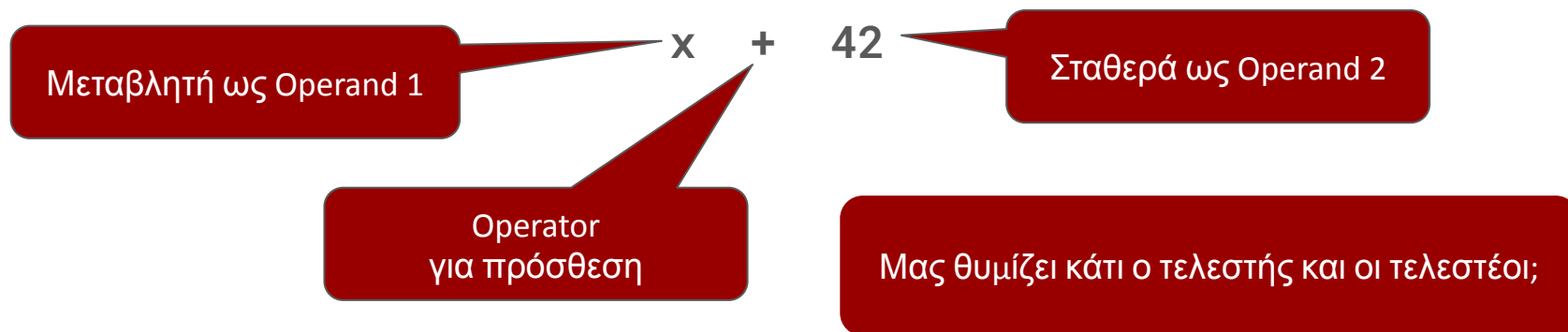
Τελεστές (Operators)

- Μοναδιαίοι (unary), Δυαδικοί (binary), Τριαδικοί (ternary)
- Αριθμητικοί, Συγκριτικοί, Λογικοί, Συνθήκης, Bitwise, Μετατροπής, Ανάθεσης
- Προτεραιότητα & Προσεταιριστικότητα

Τελεστής (Operator) και Τελεστέος (Operand)

Οι **τελεστές** στην C είναι σύμβολα που πραγματοποιούν μαθηματικούς, συγκριτικούς, δυαδικούς, υποθετικούς ή λογικούς υπολογισμούς.

Οι **τελεστέοι** στην C είναι τα αντικείμενα (μεταβλητές, σταθερές) πάνω στα οποία πραγματοποιούν τους υπολογισμούς οι τελεστές.



Πρόγραμμα - Συνάρτηση - Τελεστές

Το πρόγραμμα παίρνει κάποια δεδομένα εισόδου και υπολογίζει μια έξοδο

Το πρόγραμμα αποτελείται από συναρτήσεις που παίρνουν ορίσματα και υπολογίζουν μια έξοδο

Οι συναρτήσεις χρησιμοποιούν τελεστές και τελεστέους προκειμένου να υπολογίσουν μια έξοδο



It's turtles all the way down!

Κατηγοριοποίηση Τελεστών #1

Κατηγοριοποίηση με βάση το [arity](#) - πόσους τελεστέους έχουν:

- Μοναδιαίοι Τελεστές (Unary Operators): Ένα τελεστή (operand)
- Δυαδικοί Τελεστές (Binary Operators): Δύο τελεστές
- Τριαδικοί Τελεστές (Ternary Operators): Τρεις τελεστές

Στην συνέχεια θα δούμε κατηγοριοποίηση με βάση την χρήση του τελεστή και στην σειρά υπολογισμού

Παραδείγματα Τελεστών

Αριθμητικοί Τελεστές

Αριθμητικός Τελεστής	Ερμηνεία	Παραδείγματα
+	πρόσθεση	$40 + 2$ επιστρέφει 42 $40.0 + 2.0$ επιστρέφει 42.0
-	αφαίρεση	$44 - 2$ επιστρέφει 42 $44.0 - 2.0$ επιστρέφει 42.0
*	πολλαπλασιασμός	$42 * 2$ επιστρέφει 84 $42.0 * 2.0$ επιστρέφει 84.0
/	διαίρεση	$85.0 / 2.0$ επιστρέφει 42.5 $85 / 2$ επιστρέφει 42
%	υπόλοιπο	$7 \% 2$ επιστρέφει 1

1. Σε τι κατηγορία θα εντάσσετε τους αριθμητικούς τελεστές;
2. Τι τύπο θα είχαν οι τελεστές αν ήταν συναρτήσεις;

Συγκριτικούς Τελεστές (Comparison Operators)

Οι **συγκριτικοί τελεστές** επιτρέπουν την σύγκριση τελεστών. Το αποτέλεσμα είναι **0** (ψευδές) ή **1** (αληθές). Τι τύπου τελεστές είναι;

Σχεσιακός Τελεστής	Ερμηνεία	Παραδείγματα
==	Ίσοι τελεστές;	42 == 0 επιστρέφει 0 42 == 42 επιστρέφει 1
!=	Διαφορετικοί τελεστές;	42 != 0 επιστρέφει 1 42 != 42 επιστρέφει 0
>	Αριστερός τελεστής μεγαλύτερος του δεξιού;	42 > 0 επιστρέφει 1 42 > 42 επιστρέφει 0
<	Αριστερός τελεστής μικρότερος του δεξιού;	42 < 0 επιστρέφει 0 42 < 44 επιστρέφει 1
>=	Αριστερός τελεστής μεγαλύτερος ή ίσος του δεξιού;	42 >= 42 επιστρέφει 1 42 >= 43 επιστρέφει 0
<=	Αριστερός τελεστής μικρότερος ή ίσος του δεξιού;	42 <= 42 επιστρέφει 1 42 <= 41 επιστρέφει 0

Λογικοί Τελεστές (Logical Operators)

Οι **λογικοί τελεστές** επιτρέπουν την αποτίμηση συνδυασμού αληθών και ψευδών τιμών. Τι τύπου τελεστές είναι;

Λογικός Τελεστής	Ερμηνεία	Παραδείγματα
&&	Λογική σύζευξη (AND, \wedge) είναι και οι δύο τελεστέοι αληθείς;	$42 > 0 \ \&\& \ 2 > 3$ επιστρέφει 0 $42 > 0 \ \&\& \ 3 > 2$ επιστρέφει 1
 	Λογική διάζευξη (OR, \vee) είναι ένας εκ των δύο τελεστέων αληθής;	$2 > 3 \ \ 42 > 0$ επιστρέφει 1 $2 > 3 \ \ 42 < 0$ επιστρέφει 0
!	Λογική άρνηση (NOT, \neg) είναι ο τελεστέος ψευδής;	$!(42 < 0)$ επιστρέφει 1 $!(42 > 0)$ επιστρέφει 0

Σημαντικό: για τους λογικούς τελεστές στην C, το μηδέν (0) σημαίνει “ψευδές”. Οποιαδήποτε άλλη μη μηδενική τιμή σημαίνει “αληθές”. Το 1, το 42 και το -5 είναι όλα εξίσου “αληθή”.

Bitwise Τελεστές (Bitwise Operators)

Οι **bitwise τελεστές** κάνουν μετατροπές στα *bit* των **ακέραιων** αριθμών. Τι τύπου είναι οι τελεστές;

Bitwise Τελεστής	Ερμηνεία	Παραδείγματα
&	Σύζευξη bit (bitwise AND)	0xFF & 0xF0 επιστρέφει 0xF0 0xFF & 0x00 επιστρέφει 0x00
	Διάζευξη bit (bitwise OR)	0xFF 0xF0 επιστρέφει 0xFF 0xF0 0x0F επιστρέφει 0xFF
^	Αποκλειστική διάζευξη bit (bitwise XOR)	0xFF ^ 0xFF επιστρέφει 0x00 0x00 ^ 0xFF επιστρέφει 0xFF
~	Άρνηση bit (bitwise NOT)	~0xF0 επιστρέφει 0x0F ~0x05 επιστρέφει 0xFA
<<	Ολίσθηση bit αριστερά (shift left). Shift N bit ισοδύναμο με πολλαπλασιασμό με 2^N	2 << 4 επιστρέφει 32 4 << 1 επιστρέφει 8
>>	Ολίσθηση bit δεξιά (shift right). Shift N bit ισοδύναμο με διαίρεση με 2^N	8 >> 1 επιστρέφει 4 32 >> 4 επιστρέφει 2

Kahoot Time!

Για την Επόμενη Φορά

- Από τις σημειώσεις του κ. Σταματόπουλου συνιστώ να έχετε καλύψει τα πάντα μέχρι την σελίδα 43.
- [Type conversion](#)
- [List of operators](#)
- [lvalues and rvalues](#)
- [Precedence and associativity](#)
- Προαιρετικό: [Git Cheat Sheet](#)
- Προαιρετικό: [Git Tutorial](#)

Ευχαριστώ και καλή εβδομάδα εύχομαι!
Keep coding ;)