# Εισαγωγή στην Πληροφορική & στον Προγραμματισμό

Αρχές Προγραμματισμού Η/Υ (με τη γλώσσα C)

**Γ΄ Ανακεφαλαίωση**

Πέμπτη, 2 Ιουνίου 2022

Παναγιώτης Παύλου

c-programming-22@allos.gr

# Χρήση Debugger

Τι είναι ο debugger και ποιες είναι οι πιο κοινές λειτουργίες του

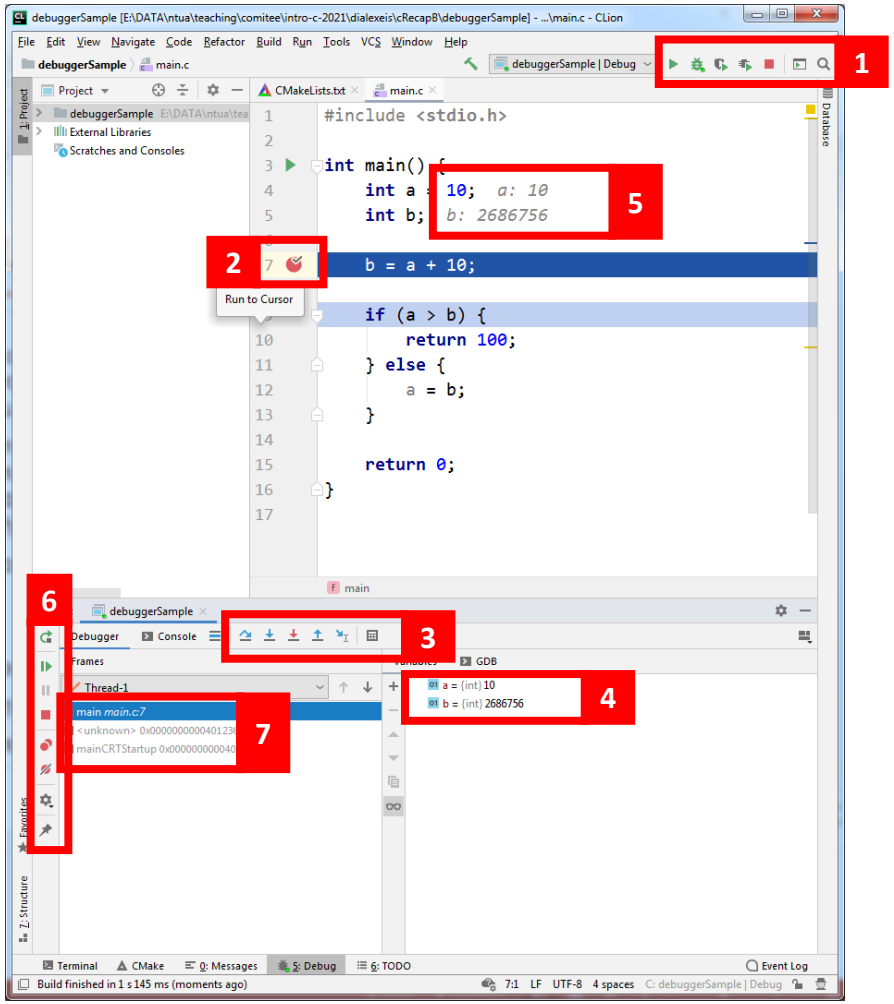
# Τι είναι ο debugger

Ο debugger επιτρέπει την ελεγχόμενη εκτέλεση ενός κώδικα από τον προγραμματιστή προκειμένου να εντοπιστούν σφάλματα (bugs) του κώδικα.

Υπάρχουν διάφορα συστήματα για το debugging της C, με τα οποία συνεργάζεται το CLion και για τα οποία παρέχει στον χρήστη του μία ενιαία διεπαφή χρήστη.

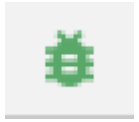
Οι κύριες λειτουργίες που παρέχονται και θα παρουσιαστούν είναι:

- τα breakpoints
- ελεγχόμενη εκτέλεση εντολή προς εντολή
- παρακολούθηση τιμών μεταβλητών
- παρακολούθηση τιμών εκφράσεων

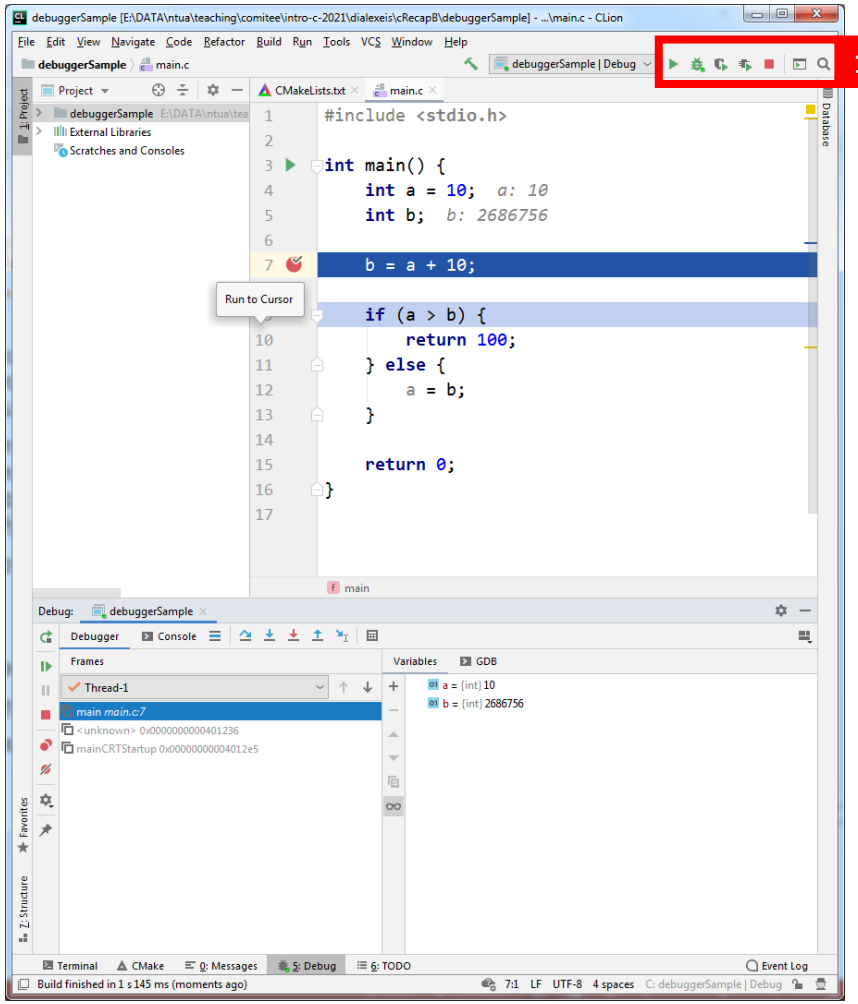


# Έναρξη της διαδικασίας debugging

Η διαδικασία του debugging ξεκινά όταν ο χρήστης επιλέξει αντί του play, το αμέσως επόμενο (2ο) εικονίδιο.



Ο τερματισμός του, γίνεται με το ίδιο εικονίδιο που σταματά και η εκτέλεση (αυτό του stop – 5ο).

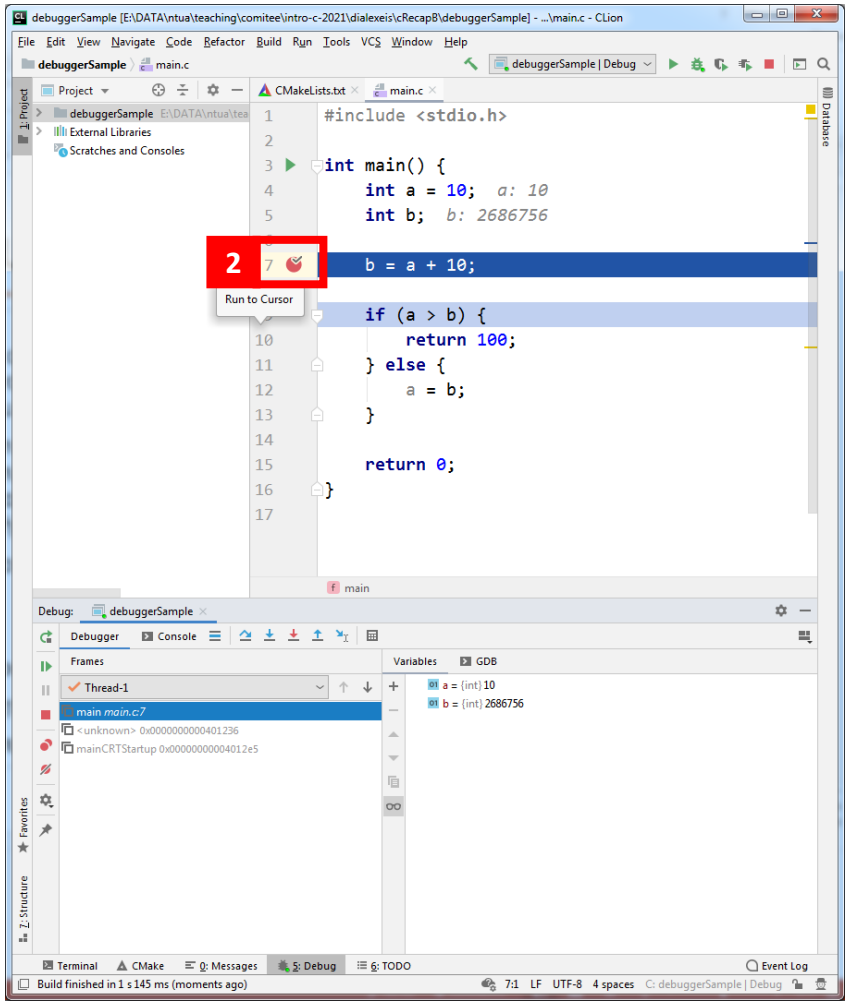


# Πως προσθέτω/αφαιρώ breakpoints

Ένα breakpoint (BP) εφαρμόζεται σε μία γραμμή που μας ενδιαφέρει να σταματήσει προσωρινά η εκτέλεση του κώδικα. Το προσθέτω ή το αφαιρώ κάνοντας κλικ στο σημείο που φαίνεται στην εικόνα , αριστερά από τη γραμμή που με ενδιαφέρει.

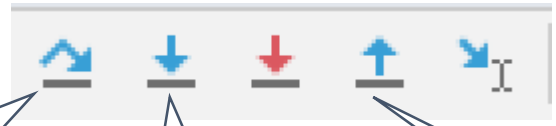
Το break point μπορεί να ισχύει:

- πάντα
- υπό συνθήκες
- μία φορά
- μόνο αφού συναντηθεί άλλο BP



# Εκτέλεση βήμα προς βήμα

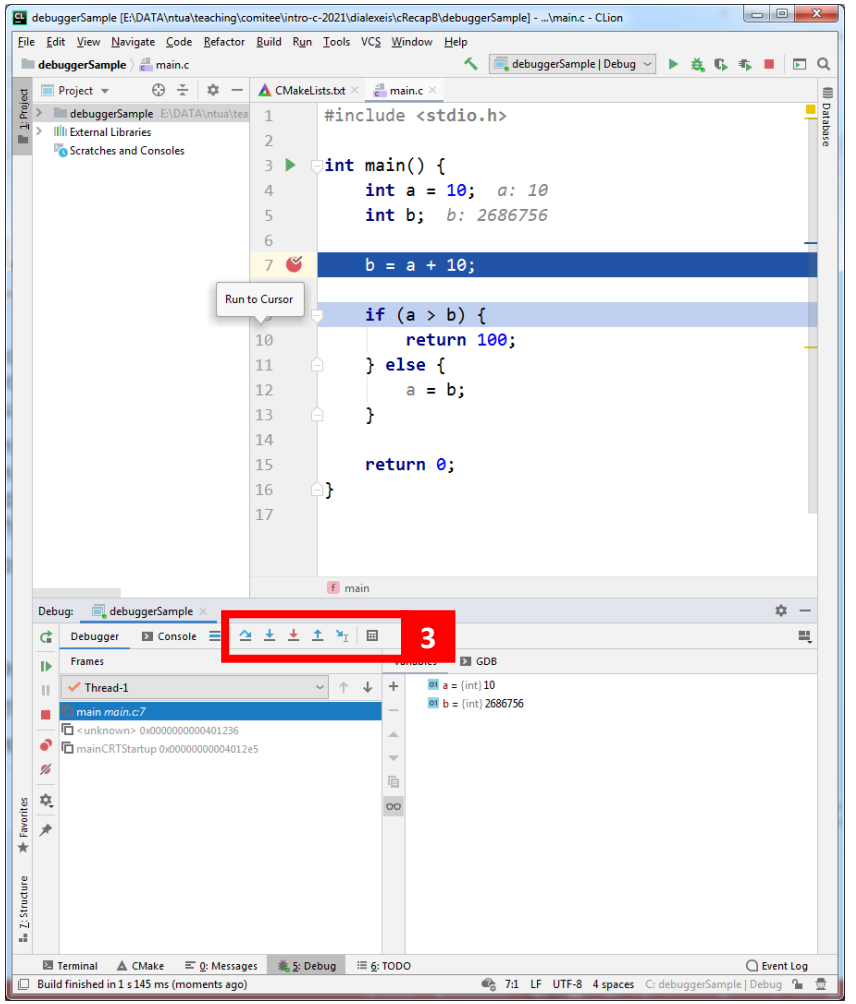
Εφόσον έχει διακοπή η εκτέλεση του κώδικα εξ'αιτίας ενός BP, μπορεί να συνεχιστεί γραμμή προς γραμμή με τα εργαλεία της μπάρας (#3).



Εκτελεί την τρέχουσα γραμμή κώδικα χωρίς να εισέρχεται αναλυτικά στις εμπλεκόμενες συναρτήσεις.

Εκτελεί την τρέχουσα γραμμή κώδικα εκτελώντας αναλυτικά τις εμπλεκόμενες συναρτήσεις.

Απρόσκοπτη συνέχιση της εκτέλεσης της τρέχουσας συνάρτησης μέχρι την ολοκλήρωσή της.

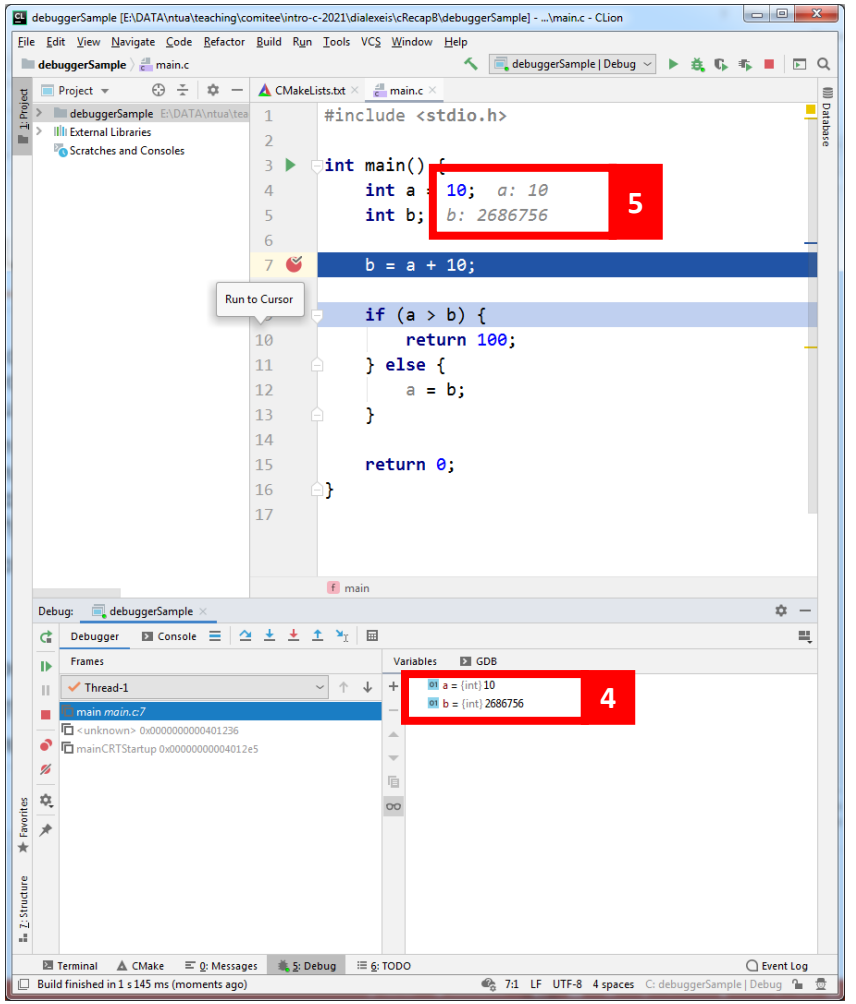


# Τιμές μεταβλητών

Όταν η εκτέλεση του κώδικα είναι σταματημένη σε κάποιο σημείο, οι τιμές των μεταβλητών εμφανίζονται σε δύο σημεία:

- Κάτω δεξιά όπου εμφανίζεται το όνομα και η τιμή των μεταβλητών (watch window)
- Μέσα στον κώδικα, δίπλα στην κάθε δήλωση της μεταβλητής με αχνά γκρι γράμματα

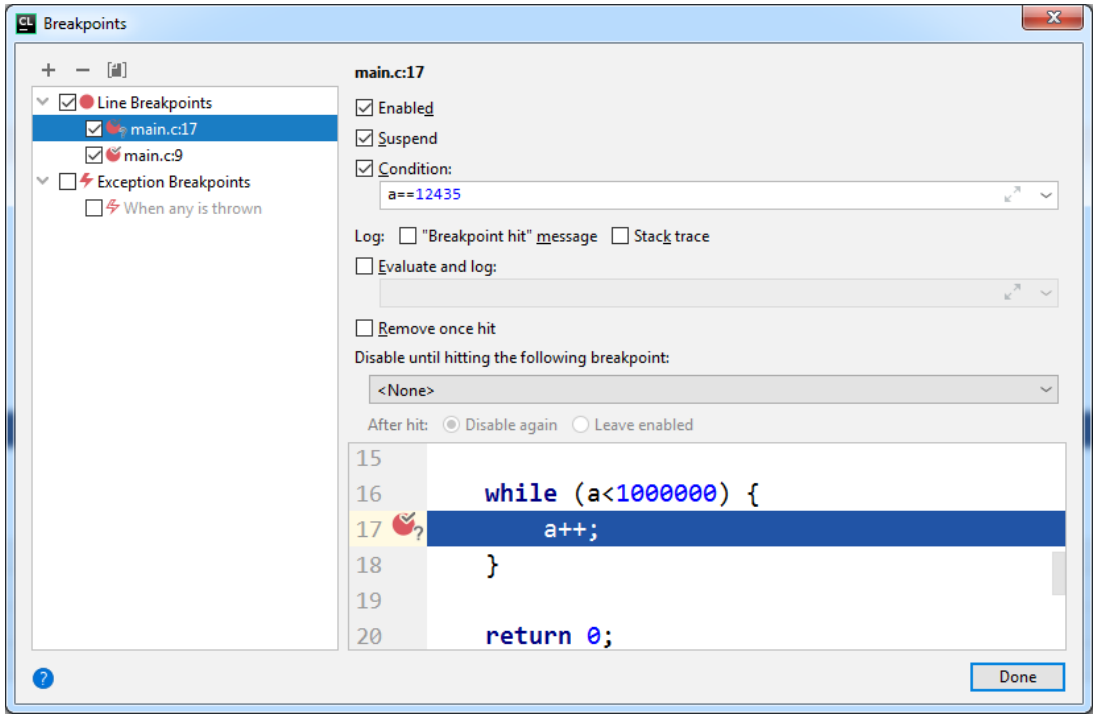
Εκτός από τις τοπικές μεταβλητές που εν γένει εμφανίζονται αυτόματα, μπορούμε να προσθέσουμε και άλλες μεταβλητές ή παραστάσεις με δεξί κλικ στο watch window.

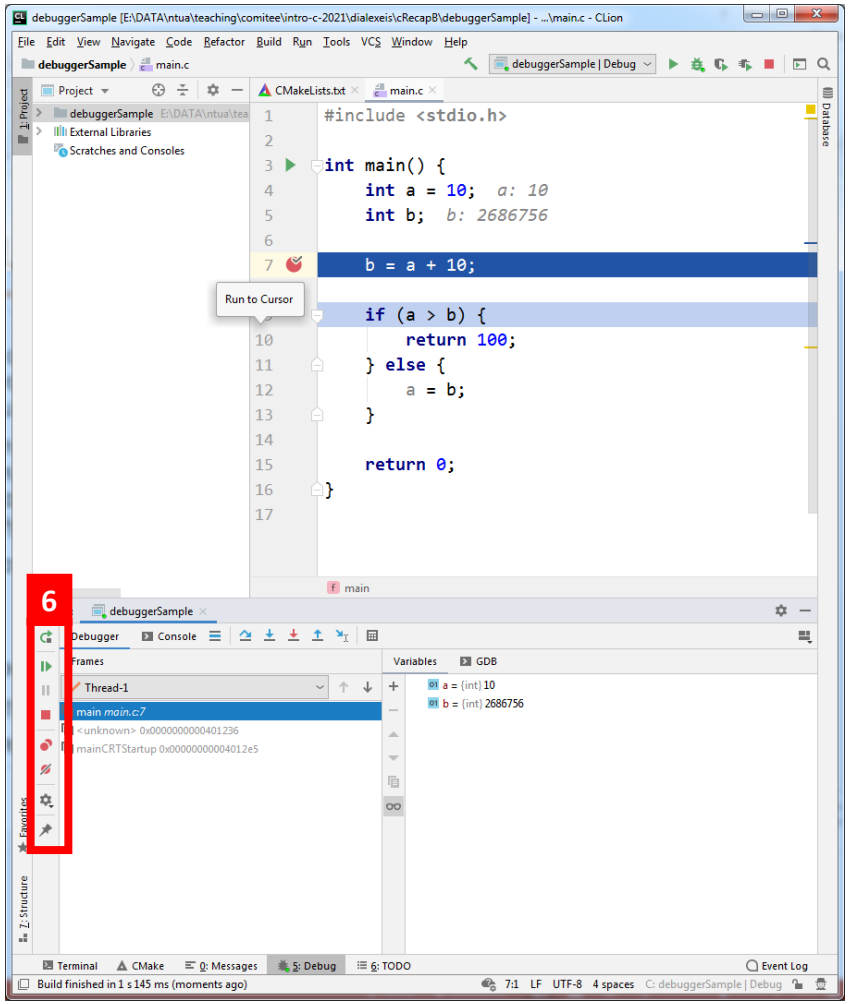


# Άλλες δυνατότητες

Διάφορες εργασίες και δυνατότητες παρέχονται στη μπάρα εργαλείων #6.

Αυτή που μας ενδιαφέρει για την παρουσίαση είναι η αναλυτική διαχείριση των BP.

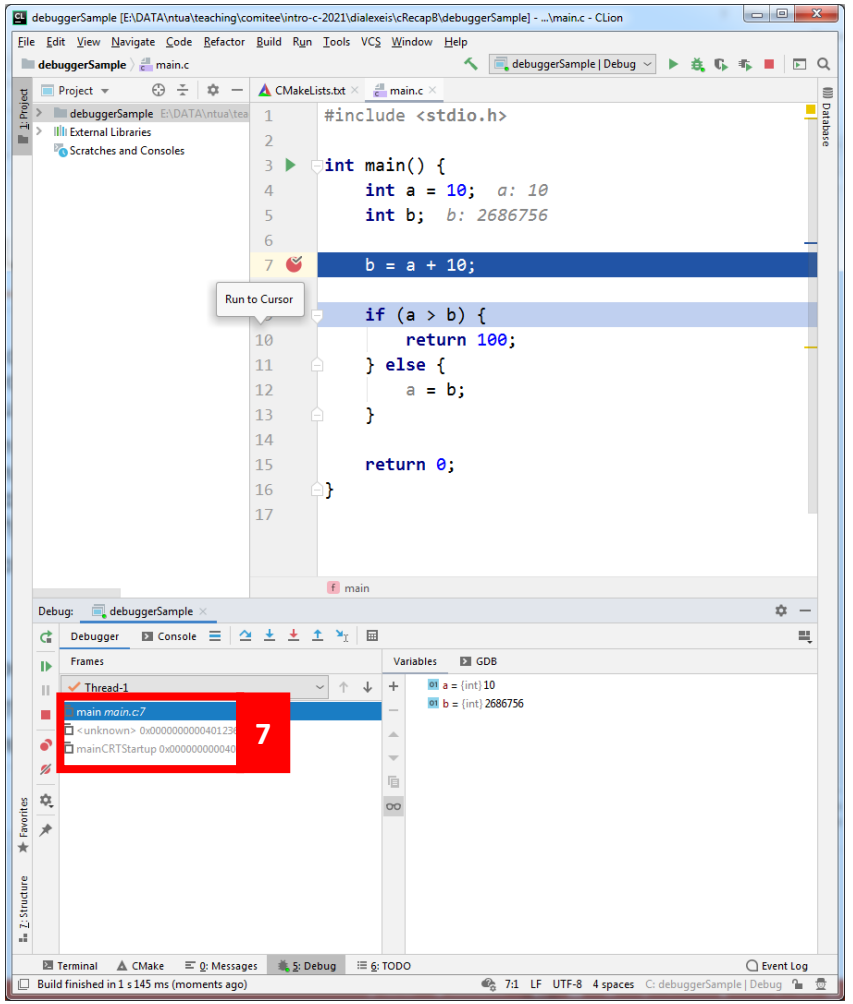


# Το call stack

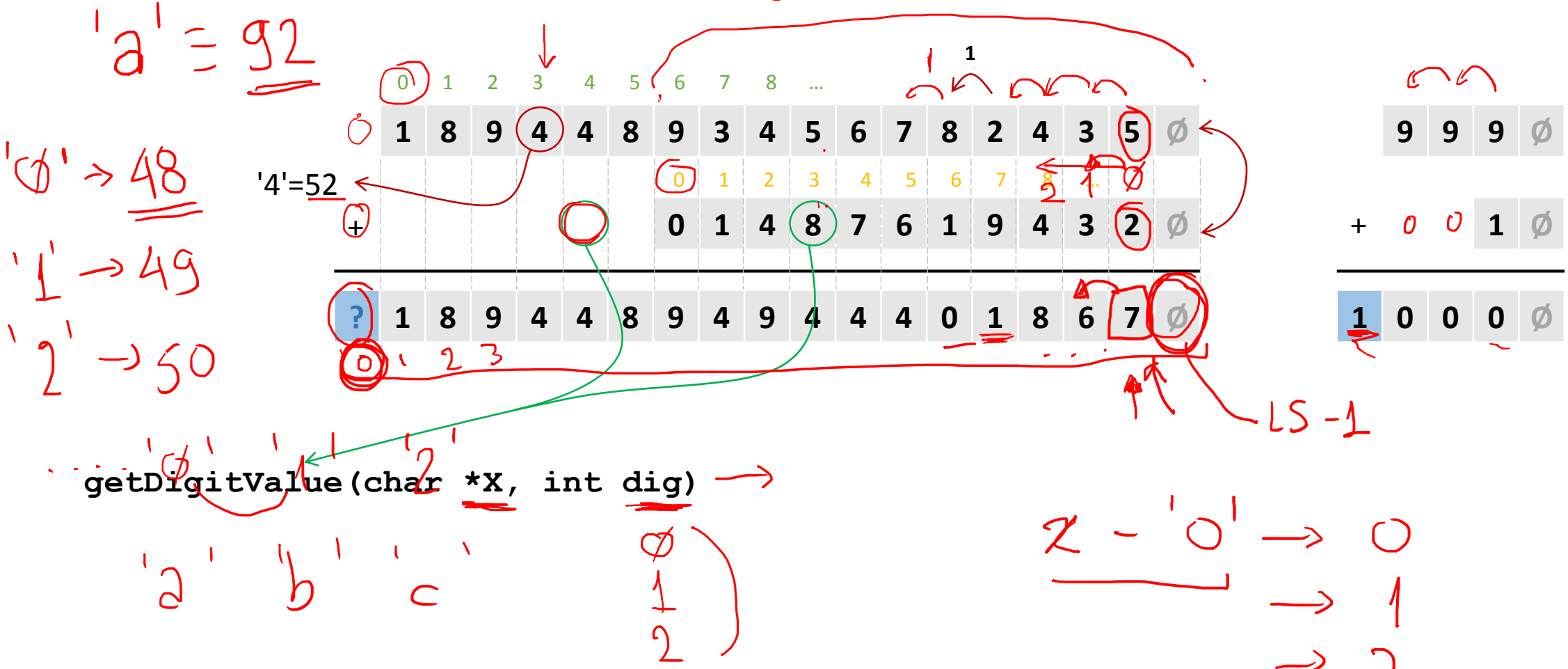
Όταν η εκτέλεση του κώδικα βρίσκεται μέσα σε κάποια συνάρτηση, την οποία έχει καλέσει κάποια άλλη, κ.ο.κ., τότε στο call stack (#7) εμφανίζονται και οι συναρτήσεις, μέσω των οποίων έχει φτάσει η εκτέλεση στο τρέχον σημείο.

Έτσι δίνεται η δυνατότητα ο χρήστης να δει με ποιες συνθήκες έχει γίνει η εκτέλεση αυτών των συναρτήσεων, επιλέγοντας την κάθε συνάρτηση στον χώρο του call stack.

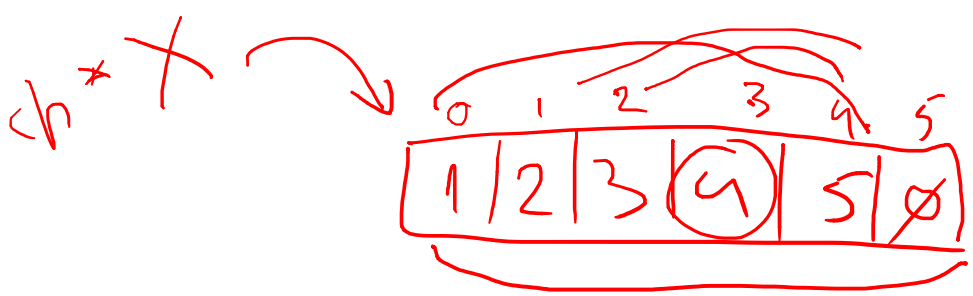


# Εφαρμογή 1 – Infinite digits math

char *sumOfIntegers(char *A, char *B)



dig  $10^{dig}$




0 → $10^0 = 1$

1 → $10^1$

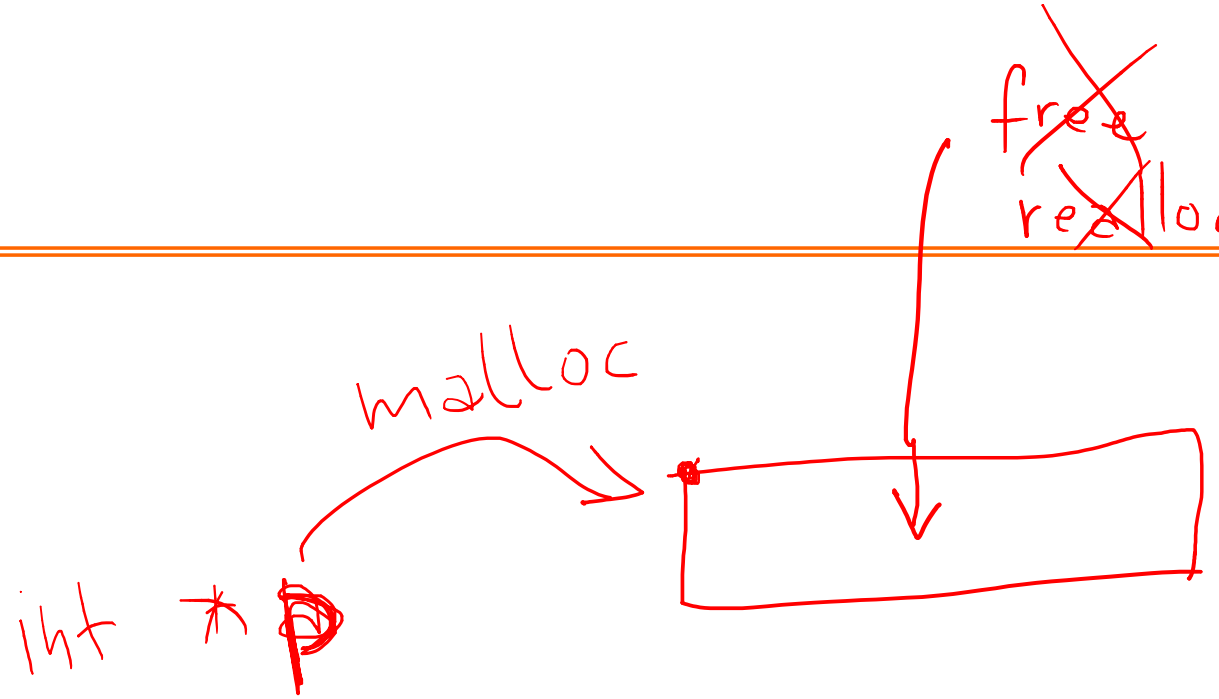
2 → $10^2$

'4' → 52

X[t - 1 - dig] - '0'

0 1 2 3 4 ..... 9

48 49 .....

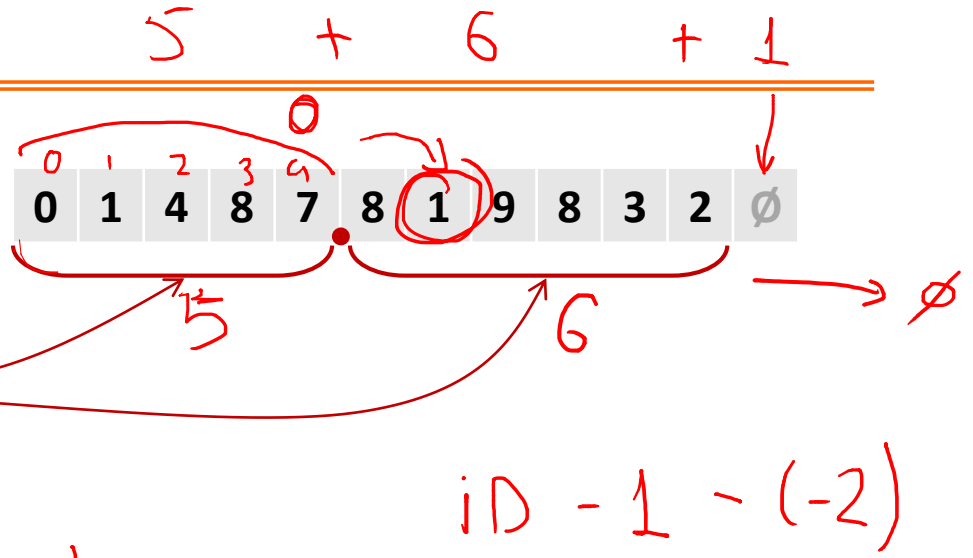



```
p += 10;
p --
p = &p[10];
```

→ free
realloc

# Εφαρμογή 2 - InfyFloat

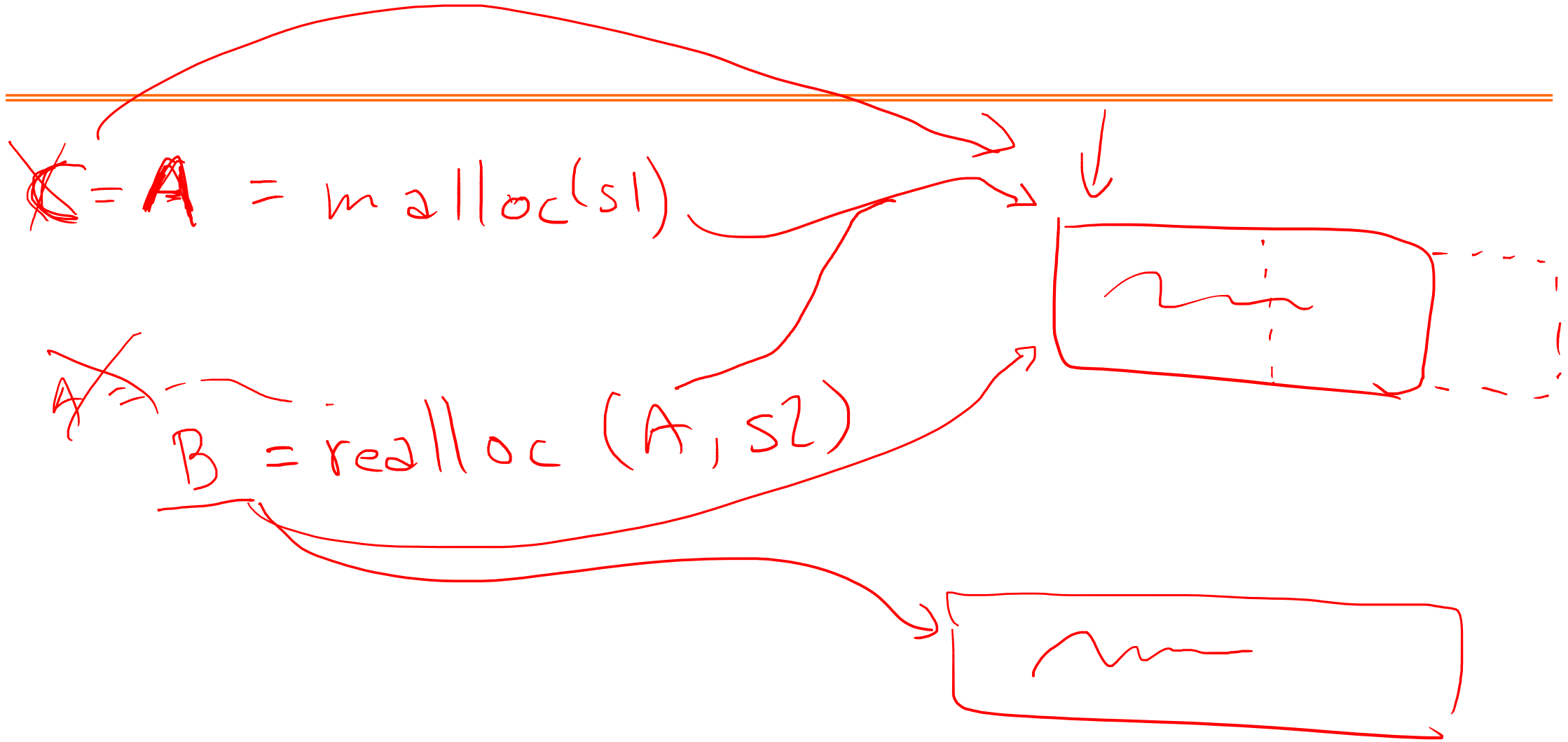
```
typedef struct _info_float {
    char *digits;
    int intDigs;
    int decDigs;
} InfyFloat;
```




```
InfyFloat *ifCreate (int iDig, int dDig)

bool ifValidate(InfyFloat *iFlo)

int ifGetDigitValue(InfyFloat *iFlo, int dig)
```


A = malloc(s1)
B = realloc (A, s2)