# Α Ανακεφαλαίωση – Μετάβαση από την C* στην C

Για την κάθε μία από τις παρακάτω εργασίες:

(α) χρησιμοποιείστε ένα ξεχωριστό project – το κατάλληλο smProject που δίνεται (για τη C)

(β) βάλτε σχόλια στον κώδικα που εξηγούν τα βήματα της επίλυσης και

(γ) δημιουργήστε μέσα στην main (ή την smMain αντίστοιχα για την C) κώδικα που θα επιδεικνύει την καλή λειτουργία των συναρτήσεων.

Σημείωση: Στον τελικό κώδικα μην χρησιμοποιήσετε τις `smPrint/printf` μέσα στις ζητούμενες συναρτήσεις.

Συμβουλές: Διαβάστε προσεκτικά δύο ή τρεις φορές την εκφώνηση. Επιλέξτε περιγραφικά και αυτοεξηγούμενα ονόματα μεταβλητών. Χρησιμοποιήστε καλή στοίχιση. Κατά τη συγγραφή του προγράμματος προσθέστε όσες `smPrint/printf` χρειαστείτε για να βλέπετε τις τιμές των μεταβλητών ώστε να εντοπίζετε ευκολότερα τυχόν λάθη, πριν παραδώσετε όμως, αφαιρέστε τις "βοηθητικές" και κρατήστε μόνο όσες είναι απαραίτητες.

**ΠΡΟΣΟΧΗ!** Μην ξεχάσετε να κατεβάσετε και να χρησιμοποιήσετε τα αντίστοιχα smProject για την κάθε άσκηση της C!

**ΠΡΟΣΕΞΤΕ ΟΠΩΣΔΗΠΟΤΕ ΤΑ ΠΑΡΑΚΑΤΩ**

Ο τρόπος με τον οποίο πρέπει να υποβάλλετε ερωτήσεις περιγράφεται εδώ:

https://qna.c-programming.allos.gr/doku.php?id=qna:technical:questions

Ο τρόπος με τον οποίο πρέπει να υποβάλλετε τον κώδικα των εργασιών στο σύστημα υποβολής περιγράφεται εδώ:

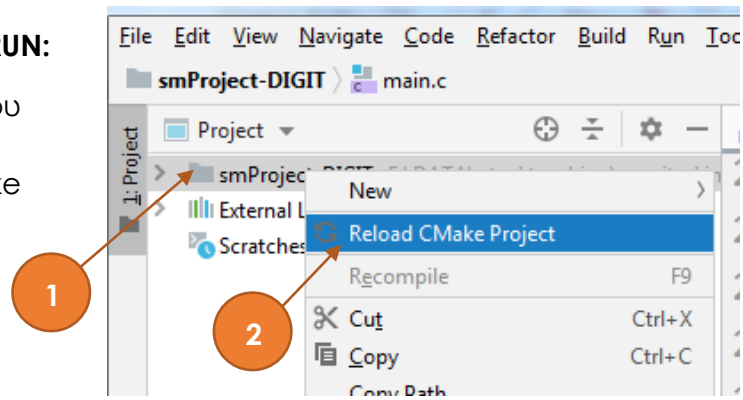
https://qna.c-programming.allos.gr/doku.php?id=qna:lesson:projects:how-to-submit

Οι έτοιμες συναρτήσεις της βιβλιοθήκης smLib που διατίθενται για χρήση στα smProjects, περιγράφονται και στο site ερωταποκρίσεων εδώ:

https://qna.c-programming.allos.gr/doku.php?id=qna:misc:sm-library

**ΕΑΝ ΔΕΝ ΕΜΦΑΝΙΖΟΝΤΑΙ ΕΝΕΡΓΑ ΤΑ BUILD / RUN:**

1. Κάνω **δεξί κλικ** πάνω στο όνομα του project και εμφανίζεται το μενού
2. Κάνω **απλό κλικ** στο Reload CMake Project, δηλαδή τη 2η επιλογή



## Εργασία $A_1$ – Full Life

Βαθμός δυσκολίας: **2/3**

*Περιγραφή*

Σε συνέχεια των προηγούμενων εργασιών με το Game of Life (δείτε εδώ) και στη μνήμη του John Conway που το επινόησε (δείτε εδώ και εδώ), καλείστε να γράψετε τις ακόλουθες συναρτήσεις.

```
function calcNextGen(rows, cols, next)
```

η οποία έχει σκοπό με βάση τα κελιά του smGrid να βρει για κάθε κελί την επόμενη κατάστασή του (την επόμενη γενιά) την οποία όμως να αποθηκεύει στο αντίστοιχο κελί του πίνακα next και την

```
function boardInit(rows, cols, next)
```

η οποία έχει σκοπό να ανάβει τα κελιά του smGrid (αφού το αρχικοποιήσει στις διαστάσεις που δίνονται) όπου το αντίστοιχο στοιχείο του next είναι αληθές.

Και οι δύο συναρτήσεις, εφόσον ολοκληρωθεί επιτυχώς η εκτέλεσή τους θα πρέπει να επιστρέφουν αληθές, ενώ εάν ολοκληρωθούν με σφάλμα να επιστρέφουν ψευδές. Προσέξτε ότι η **calcNextGen** θα πρέπει να ελέγχει ότι συμπίπτουν οι διαστάσεις του πίνακα next (rows & cols) με αυτές του smGrid και να επιστρέφει ψευδές όταν δεν συμπίπτουν.

**ΣΗΜΕΙΩΣΗ :** Οι συναρτήσεις nextGen και countNeighbors υπάρχουν στη σελίδα της απλοποιημένης C (χωρίς να φαίνονται) και θα πρέπει να χρησιμοποιήσετε αυτές αντί των δικών σας που είχατε παραδώσει στις εργασίες 3α και 4α.

**ΠΡΟΣΟΧΗ :** Η κατάσταση της επόμενης γενιάς δεν μπορεί και δεν πρέπει να συμπληρωθεί από το smGrid πάνω στο smGrid, έτσι χρειάζεται η χρήση του βοηθητικού πίνακα next.

**ΣΟΣ** : Η παράμετρος **next** είναι πίνακας με **rows** γραμμές και **cols** στήλες και στις δύο συναρτήσεις. Στην 1η όμως λειτουργεί ως «έξοδος», δηλαδή τη συμπληρώνει η συνάρτηση, ενώ στη 2η ως «είσοδος».

## Εργασία $A_2$ – Αλγεβρικοί πίνακες

Βαθμός δυσκολίας: **1/3**

Όνομα smProject: **smProject-MATRIX-1**

*Περιγραφή*

Καλείστε να γράψετε τις ακόλουθες συναρτήσεις:

```
bool isLTriangular(int R, int C, double A[R][C])
```

η οποία επιστρέφει αληθές όταν ο πίνακας A είναι **κάτω τριγωνικός** δηλαδή τα στοιχεία πάνω και δεξιά από την κύρια διαγώνιο είναι όλα μηδενικά και ο A είναι τετραγωνικός πίνακας, αλλιώς επιστρέφει ψευδές. Επίσης γράψτε την

```
bool add(int R, int C, double A[R][C], double B[R][C] , double sum[R][C])
```

η οποία επιστρέφει αληθές όταν **προσθέτει** επιτυχώς δύο αλγεβρικούς πίνακες A και B διαστάσεων R x C και αποθηκεύει το αποτέλεσμα στον sum (RxC), αφήνοντας άθικτους τους A και B. Και την

```
bool mul(int M, int N, int K, double A[M][N], double B[N][K] , double P[M][K])
```

η οποία επιστρέφει αληθές όταν **πολλαπλασιάζει** επιτυχώς δύο αλγεβρικούς πίνακες A (MxN) και B (NxK) και αποθηκεύει το αποτέλεσμα στον P (MxK), αφήνοντας άθικτους τους A και B.

Επιπλέον, εάν δοθούν παράλογες διαστάσεις στους πίνακες, (δηλαδή μηδενικές ή αρνητικές) τότε θα πρέπει να επιστρέφουν ψευδές όλες οι συναρτήσεις χωρίς να επιτελούν κάποια άλλη εργασία.