4η διάλεξη – βρόχοι for & while, do/while & switch, τελεστές ανάθεσης

Για την κάθε μία από τις παρακάτω εργασίες:

(α) χρησιμοποιείστε ένα ξεχωριστό project – το κατάλληλο smProject που δίνεται,

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

(γ) δημιουργήστε μέσα στην smMain κώδικα που θα επιδεικνύει την καλή λειτουργία των συναρτήσεων.

Σημείωση: Στον τελικό κώδικα μην χρησιμοποιήσετε τις `printf`, `smGrid`, `smShowGrid`, `smGrid2Image` 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 `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.

Σημείωση: Στον τελικό κώδικα μην χρησιμοποιήσετε την `printf` μέσα στις ζητούμενες συναρτήσεις.

Συμβουλές: Διαβάστε προσεκτικά την εκφώνηση. Επιλέξτε περιγραφικά ονόματα μεταβλητών. Χρησιμοποιήστε καλή στοίχιση (formatting) του κώδικα.

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

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:

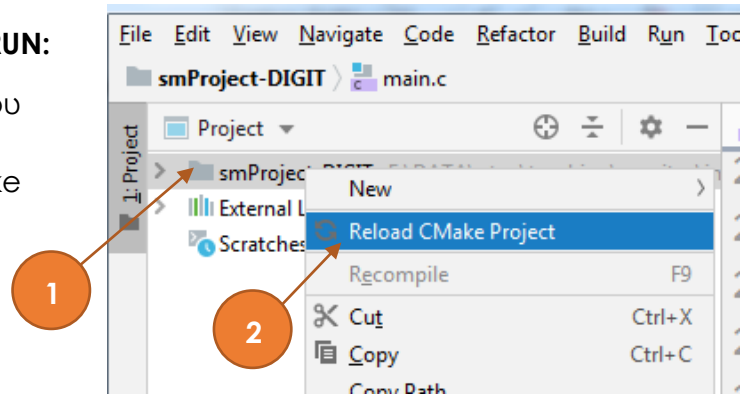
https://qna.c-programming.allos.gr/doku.php?id=qna:lesson:projects:how-to-submit

Οι έτοιμες συναρτήσεις της βιβλιοθήκης smLib που διατίθενται για χρήση στα smProjects, περιγράφονται (εκτός από τις διαφάνειες) 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 4α – Παραγοντικό

Βαθμός δυσκολίας: **1/3**

Όνομα smProject: **smProject-FACTORIAL**

Περιγραφή

Ο υπολογισμός παραγοντικού ενός ακέραιου αριθμού μεγαλύτερου ή ίσου από το 1, είναι γνωστό το πώς υπολογίζεται. Γράψτε λοιπόν τη συνάρτηση:

```
unsigned long long factorial(unsigned char number)
```

Η οποία επιστρέφει το παραγοντικό του δεδομένου αριθμού `number`.

Σημειώστε ότι το παραγοντικό του 0 είναι ίσο με 1.

Παρατηρείστε μέχρι ποιόν αριθμό μπορείτε να παραστήσετε το παραγοντικό του, με `unsigned long long` τύπο δεδομένων, και συμπληρώστε τον στο πεδίο «Σχόλιο» στο σύστημα υποβολών. Εντυπωσιακό, ε;!

Εργασία 4β – Game of life , μέτρηση γειτόνων

Βαθμός δυσκολίας: **1/3**

Ονόμα smProject: **smProject-LIFE-2**

Περιγραφή

Το Game of life (https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life) το γνωρίσαμε σε προηγούμενη άσκηση.

Στην άσκηση εκείνη θεωρούνταν ήδη υπολογισμένο το πλήθος των γειτονικών κελιών που είναι ζωντανά για κάθε κελί.

Με τη χρήση του grid της smLib δημιουργήστε μία συνάρτηση:

```
int countNeighbors ( int x, int y )
```

η οποία επιστρέφει το πλήθος των ζωντανών γειτόνων του κελιού x,y. Εάν το κελί x,y είναι εκτός του πλέγματος τότε πρέπει να επιστρέφει -1, ως ένδειξη λάθους.

Προσέξτε όμως τα κελιά που είναι στο όριο του πλέγματος, καθώς μερικά από τα γειτονικά τους είναι εκτός πλέγματος, αλλά δεν πρέπει βέβαια να επιστρέφεται -1 για αυτά.

Από την smLib θα χρειαστείτε τις συναρτήσεις: **`smgTest`**, **`smgWidth`**, **`smgHeight`**.

Στην main εάν θέλετε μπορείτε να φτιάξετε ένα πλέγμα να ανάψετε κάποια κελιά για να κάνετε τις δικές σας δοκιμές. Όμως στις ενσωματωμένες δοκιμές υπάρχουν αρκετές περιπτώσεις στις οποίες μπορεί – αν και δεν πρέπει – ο κώδικάς σας να αποτύχει.

Εργασία 4η – Οριζόντιες και κατακόρυφες

Βαθμός δυσκολίας: **3/3**

Όνομα smProject: **smProject-LINES-VH**

Περιγραφή

Καλείστε να γράψετε τις συναρτήσεις:

- `bool` **`drawVLine`**`(int x, int y1, int y2)` αυτή πρέπει να σχεδιάζει στο grid μία κατακόρυφη γραμμή από το σημείο (x,y1) μέχρι και το (x,y2)
- `bool` **`drawHLine`**`(int y, int x1, int x2)` αυτή πρέπει να σχεδιάζει στο grid μία οριζόντια γραμμή από το σημείο (x1,y) μέχρι και το (x2,y)
- `bool` **`drawALine`**`(int x, int y, int L, bool isV)` αυτή πρέπει να σχεδιάζει μία γραμμή που ξεκινά από το (x,y) και εκτείνεται δεξιά ή κάτω (αν το isVert είναι false ή true αντίστοιχα) με συνολικό μήκος L κελιά. Φυσικά το L πρέπει να είναι θετικό.

Και οι 3 συναρτήσεις θα πρέπει να επιστρέφουν true εάν όλα πήγαν καλά, ενώ false εάν το ευθύγραμμο τμήμα που θα σχεδιάσουν βγαίνει έξω από τα όρια του grid, ή όταν το L είναι αρνητικό.

Στις υπόλοιπες περιπτώσεις θα πρέπει να λειτουργεί.

Η **`drawALine`** θα πρέπει να χρησιμοποιεί της **`drawHLine`** και **`drawVLine`** , ώστε να είναι απλή κατά το δυνατόν.

ΣΗΜΕΙΩΣΗ: Εάν για κάποιο λόγο δεν υλοποιηθεί κάποια από αυτές τις συναρτήσεις μπορείτε να την ορίσετε και να κάνει απλά "return true;" ώστε να γίνει compile ο κώδικας.