ΣΕΤ ΑΣΚΗΣΕΩΝ 3

ΕΡΓΑΣΤΗΡΙΟ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ Ι, ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2018-2019

Προθεσμία: Παρασκευή 11/1/2019, 22:00

Διαβάστε πριν ξεκινήσετε

Διαβάστε την εκφώνηση προσεκτικά και "σχεδιάστε" το πρόγραμμά σας στο χαρτί.

Για κάθε στάδιο, αποφασίστε τι μεταβλητές θα χρειαστείτε, τι ονόματα θα τους δώσετε, αν χρειάζονται σταθερές κι αν ναι για ποιες ποσότητες, τι δομές ελέγχου θα χρησιμοποιήσετε για κάθε λειτουργία και πώς θα κάνετε τους υπολογισμούς που χρειάζονται.

Μη διστάζετε να ζητήσετε βοήθεια! Χρησιμοποιήστε κατά προτίμηση την εφαρμογή συζητήσεων στο e-class και μόνο αν είναι απαραίτητο email στο `ce120lab@gmail.com` (π.χ. αν πραγματικά επιβάλλεται να στείλετε κάποιο κομμάτι κώδικα μαζί με το μήνυμά σας).

Η εργασία αυτή μπορεί να γίνει σε ομάδες μέχρι και 2 ατόμων. Δε χρειάζεται να είστε ομάδα με το ίδιο άτομο που είστε στο εργαστήριο ή το ίδιο άτομο με το οποίο κάνατε την 1η εργασία. Μπορείτε να συζητάτε τις ασκήσεις με συμφοιτητές σας αλλά δεν επιτρέπεται η ανταλλαγή κώδικα με οποιονδήποτε τρόπο.

Ξεκινήστε νωρίς! Ο προγραμματισμός είναι πάντα ΠΟΛΥ πιο χρονοβόρος από ότι περιμένετε.

Εκπρόθεσμες ασκήσεις δε γίνονται δεκτές.

Οι ασκήσεις σας θα βαθμολογηθούν στα παρακάτω (χωρίς ιδιαίτερη σειρά):

- Ορθότητα
- Γενική μορφοποίηση προγράμματος (στοίχιση, ονόματα μεταβλητών και σταθερών, κτλ.)
- Σχεδιασμός προγράμματος και αποτελεσματική χρήση κατάλληλων δομών, μεταβλητών, σταθερών κτλ.
- Σχεδιασμός και χρήση κατάλληλων συναρτήσεων. Θα σας προτείνουμε κάποιες, αλλά περιμένουμε να σκεφτείτε και δικές σας.
- Συμμόρφωση με τις προδιαγραφές
- Αποτελεσματικά σχόλια, σύμφωνα με τους κανόνες σχολιασμού του σχετικού φυλλαδίου.

Απαγορεύεται αυστηρά η χρήση goto, η χρήση gets και η χρήση καθολικών μεταβλητών.

ΔΙΑΒΑΣΤΕ ΟΛΗ ΤΗΝ ΕΚΦΩΝΗΣΗ ΚΑΘΕ ΑΣΚΗΣΗΣ ΠΡΙΝ ΞΕΚΙΝΗΣΕΤΕ!

Άσκηση 1: Επεξεργασία εικόνας

Στην παρούσα άσκηση θα γράψετε ένα πρόγραμμα επεξεργασίας εικόνας. Μπορείτε να φανταστείτε μία εικόνα σαν ένα διδιάστατο πίνακα από εικονοστοιχεία (στην αγγλική βιβλιογραφία αναφέρονται ως *pixels*), μεγέθους όσο και το μέγεθος της εικόνας. Για παράδειγμα, μία εικόνα 712x512 pixels αντιστοιχεί σε ένα διδιάστατο πίνακα από pixels 712 στηλών και 512 γραμμών (συνηθίζουμε να αναφέρουμε πρώτα το πλάτος και μετά το ύψος της εικόνας).

Κάθε pixel απεικονίζει τις τιμές για τα τρία βασικά χρώματα κόκκινο, πράσινο και μπλε, γνωστά και ως RGB από τα αρχικά των λέξεων *red, green, blue* στην αγγλική γλώσσα. Θεωρούμε ότι οι τιμές αυτές είναι πάντα φυσικοί αριθμοί στο εύρος [0, 255].

Παραδείγματα **RGB** τιμών είναι τα εξής:

1. ένα κόκκινο pixel έχει RGB τιμές 255, 0, 0 (μέγιστη τιμή για το κόκκινο και μηδέν για τα υπόλοιπα).
2. Ένα κίτρινο pixel προκύπτει από τις RGB τιμές 255, 255, 0 (το κίτρινο προκύπτει από τη μίξη του κόκκινου με το πράσινο χρώμα)
3. Ένα άσπρο pixel προκύπτει από τις RGB τιμές 255, 255, 255 (μέγιστη τιμή για όλα τα χρώματα)
4. Ένα μαύρο pixel προκύπτει από τις RGB τιμές 0, 0, 0 (ελάχιστη τιμή για όλα τα χρώματα)

Μπορείτε να δοκιμάσετε διαφορετικούς χρωματικούς συνδυασμούς εδώ ή με χρήση κάποιου προγράμματος επεξεργασία εικόνας (π.χ. gimp).

Η μορφή της εικόνας PPM

Στην παρούσα εργασία θα χρησιμοποιήσετε για διάβασμα από αρχείο και αποθήκευση σε αρχείο εικόνες που είναι αποθηκευμένες σε μορφή (format) PPM. Πρόκειται για έγχρωμες εικόνες που η πληροφορία της εικόνας αποθηκεύεται σε μορφή κειμένου. Η μορφή ενός αρχείου PPM έχει ως εξής:

1. Ξεκινάει με το αλφαριθμητικό **`P3`**.
2. Ακολουθεί ένας ακέραιος που αντιστοιχεί στο πλάτος της εικόνας (σε pixels).
3. Ακολουθεί ένας ακέραιος που αντιστοιχεί στο ύψος της εικόνας (σε pixels).
4. Ακολουθεί ένας ακέραιος που αντιστοιχεί στη μέγιστη τιμή που μπορεί να πάρει ένα χρώμα στην εικόνα (για τις δικές μας εικόνες η μέγιστη τιμή θα είναι 255).

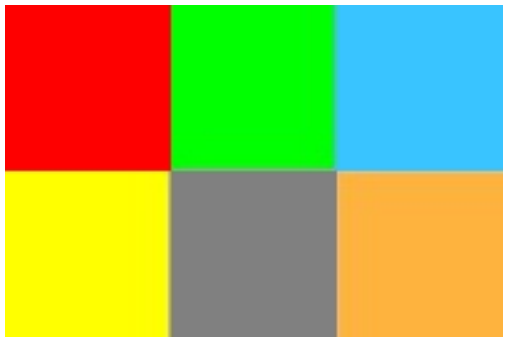
   Τα σημεία 1-4 αποτελούν την κεφαλίδα του αρχείου.
5. Στη συνέχεια για κάθε pixel εμφανίζονται 3 ακέραιοι για τα τρία **RGB** χρώματα κόκκινο, πράσινο και μπλε με τη σειρά. Το αρχείο ξεκινά με την πληροφορία του επάνω αριστερού pixel (θεωρούμε ως πρώτη σειρά pixels, την κορυφαία σειρά), έπεται το αμέσως δεξιότερο pixel, μέχρι να φτάσουμε στο δεξιότερο στοιχείο της κορυφαίας σειράς. Στη συνέχεια το αρχείο συνεχίζει από το αριστερότερο στοιχείο της επόμενης σειράς κ.ο.κ.

   Για παράδειγμα, εάν έχουμε μία εικόνα μεγέθους **25x25** pixels στο αρχείο θα έχουμε αποθηκευμένους **25x25x3** ακεραίους που αντιστοιχούν στην πληροφορία των pixels της εικόνας. Η τιμή κάθε ακεραίου δεν πρέπει να υπερβαίνει τη μέγιστη τιμή που ορίστηκε στην αρχή του αρχείου.

Οποιοδήποτε αλφαριθμητικό εντός του αρχείου διαχωρίζεται από την υπόλοιπη πληροφορία με έναν ή περισσότερους κενούς χαρακτήρες, χαρακτήρες tab ή χαρακτήρες αλλαγής γραμμής ή συνδυασμούς των παραπάνω.

Παρακάτω δίνεται ένα παράδειγμα εικόνας PPM μεγέθους 2x3 και δίπλα η εικόνα που αντιστοιχεί σε αυτό σε μεγέθυνση. Το παρακάτω περιεχόμενο αντιστοιχεί στο αρχείο κειμένου **test.ppm** που σας παρέχεται έτοιμο για τον έλεγχο του προγράμματος σας.

```
P3
3 2
255
255   0   0   0 255   0  58 196 255
255 255   0 128 128 128 255 179  63
```



Διάβασμα της εικόνας και αποθήκευση σε πίνακα

Κατ' αρχήν απαραίτητη προϋπόθεση για το διάβασμα της πληροφορίας σε πίνακα είναι ο πίνακας να είναι ικανού μεγέθους, ώστε να μπορεί να αποθηκεύσει την εικόνα. Ο πίνακας μπορεί να έχει διαστάσεις μεγαλύτερες ή ίσες της εικόνας που πρόκειται να αποθηκεύσει, όχι όμως μικρότερες.

Εφόσον ισχύει η παραπάνω προϋπόθεση, το διάβασμα μιας εικόνας και η αποθήκευση σε διδιάστατο πίνακα γίνεται ως εξής. Η πρώτη σειρά pixels όπως είναι αποθηκευμένη στο αρχείο αποθηκεύεται στην πρώτη γραμμή του πίνακα με την σειρά που αυτά διαβάζονται, η δεύτερη σειρά pixels στη δεύτερη γραμμή του πίνακα κ.ο.κ.

Η αποθήκευση της εικόνας από τον πίνακα σε αρχείο γίνεται ως εξής. Αρχικά γράφεται η επικεφαλίδα της εικόνας, δηλαδή η συμβολοσειρά P3, το πλάτος, το ύψος και η μέγιστη τιμή χρώματος. Στη συνέχεια καταγράφονται οι τιμές RGB όλων των pixels ανά γραμμή ξεκινώντας από το 1ο στοιχείο της 1η γραμμής του πίνακα.

Για κάθε pixel διατηρούμε τρία χρώματα, επομένως ένας τρόπος να αποθηκεύσουμε την πληροφορία είναι με χρήση struct τριών πεδίων. Εφόσον οι τιμές των pixels για κάθε βασικό RGB χρώμα κυμαίνονται μεταξύ 0 και 255, οποιαδήποτε τιμή χωράει σε 1 byte. Γι αυτό το λόγο αρκεί ο τύπος κάθε πεδίου του struct να είναι `unsigned char`.

Έλεγχος του προγράμματος

Για να ελέγξετε το πρόγραμμα σας δίνονται οι εξής εικόνες:

1. **test.ppm** μεγέθους 2x3. Στοιχειώδης εικόνα για την εύκολη αποσφαλμάτωση του κώδικα σας.
2. **lena.ppm** μεγέθους 512x512.
3. **landscape.ppm** μεγέθους 500x314.

Όλες τις εικόνες PPM μπορείτε να τις ανοίξετε για να δείτε το περιεχόμενο τους με χρήση του προγράμματος kate ή οποιουδήποτε άλλου text editor.

Ανάπτυξη του προγράμματος

Βήμα 1 – Δήλωση μεταβλητών

Μελετήστε το αρχείο `hw3a.h` μέσα στο οποίο ορίζεται η σταθερά **`IMG_SIZE`**, ορίζεται ένα struct που αναπαριστά ένα pixel και δηλώνονται οι συναρτήσεις **`load_image`**, **`save_image`** οι υλοποιήσεις των οποίων σας παρέχονται στη στατική βιβλιοθήκη **`libhw3.a`**.

Το `hw3a.h` γίνεται ήδη #include στο αρχείο `hw3a.c` και μπορείτε να χρησιμοποιήσετε τα περιεχόμενά του στο πρόγραμμά σας χωρίς να κάνετε κάτι άλλο.

Στη συνάρτηση **`main`** καλείστε να δηλώσετε τις εξής μεταβλητές:

1. ένα διδιάστατο πίνακα από struct pixel μεγέθους **`IMG_SIZE`**×**`IMG_SIZE`**
2. Έναν ακέραιο που αποθηκεύει το πλάτος της εικόνας σε pixels.
3. Έναν ακέραιο που αποθηκεύει το ύψος της εικόνας σε pixels.
4. Έναν ακέραιο που αποθηκεύει τη μέγιστη τιμή χρώματος για την εικόνα

Βήμα 2 – Διάβασμα της εικόνας από αρχείο

Στη συνάρτηση **`main`** γράψτε κώδικα, ώστε όταν το πρόγραμμα ξεκινά, να εμφανίζει **χαρακτήρα αλλαγής γραμμής και** το μήνυμα "**`Enter image path:`** " με ένα κενό μετά το ':'. Στη συνέχεια περιμένει από το χρήστη να εισάγει ένα έγκυρο file path από το οποίο θα διαβάσει μία εικόνα. Το file path που θα δώσει ο χρήστης μπορεί να είναι απόλυτο (π.χ. /home/myusername/images/test.ppm) ή σχετικό ως προς την θέση που βρίσκεται (π.χ. images/test.ppm), υποθέτοντας ότι βρίσκεστε μέσα στον κατάλογο /home/myusername.

Αφού διαβάσει το file path που δίνει ο χρήστης επιχειρεί να διαβάσει την εικόνα από το αρχείο με τη βοήθεια της συνάρτησης **`load_image`**.

Η `load_image` παίρνει ως παραμέτρους το file path της εικόνας, έναν αρχικά άδειο πίνακα από pixels δύο διαστάσεων στον οποίο θα αποθηκεύσει τις τιμές των χρωμάτων της εικόνας που θα διαβάσει από το αρχείο, τη διεύθυνση ενός ακεραίου όπου θα αποθηκεύσει το πλάτος της εικόνας, τη διεύθυνση ενός ακεραίου όπου θα αποθηκεύσει το ύψος της εικόνας και τη διεύθυνση ενός ακεραίου όπου θα αποθηκεύσει τη μέγιστη τιμή χρώματος στην εικόνα. Όλες αυτές οι πληροφορίες θα παρθούν από το αρχείο. Χρησιμοποιούμε τις διευθύνσεις των ακεραίων για τις διαστάσεις και το μέγιστο χρώμα ώστε να μπορούμε να "επιστρέψουμε" έμμεσα αυτές τις τιμές από τη συνάρτηση `load_image` στη συνάρτηση που την καλεί. Η `load_image` επιστρέφει 1 αν αποθηκεύσει επιτυχώς την εικόνα στον πίνακα και 0 αν αποτύχει.

Εάν η κλήση στη `load_image` αποτύχει, το πρόγραμμα εκτυπώνει χαρακτήρα αλλαγής γραμμής και το μήνυμα "**`Invalid path!`**" ακολουθούμενο από χαρακτήρα αλλαγής γραμμής κι επαναλαμβάνει τη διαδικασία μέχρι ο χρήστης να δώσει ένα έγκυρο path. Μετά το επιτυχημένο διάβασμα της εικόνας εκτυπώνει χαρακτήρα αλλαγής γραμμής και το μήνυμα "**`Load OK!`**" ακολουθούμενο **από το διαχωριστικό string.**

Σημείωση: Συνιστάται να διαβάσετε το file path με χρήση της συνάρτησης **`fgets`** αντί για **`scanf`**. Η **`fgets`** έχει το πλεονέκτημα ότι διαβάζει συμβολοσειρές που περιέχουν κενά, σε αντίθεση με τη **`scanf`** που τυπικά θεωρεί τον κενό χαρακτήρα, ως χαρακτήρα τερματισμού της τρέχουσας ανάγνωσης. Λάβετε υπόψη ότι η **`fgets`** αποθηκεύει στη συμβολοσειρά που διάβασε και το '\n' που πληκτρολόγησε ο χρήστης. Δείτε περισσότερα στο παράρτημα στο τέλος της εκφώνησης.

Βεβαιωθείτε πως έχει γίνει σωστά η ανάγνωση και αποθήκευση της εικόνας πριν προχωρήσετε στο επόμενο βήμα.

Βήμα 3 – Εκτύπωση του περιεχομένου της εικόνας στην οθόνη

Γράψτε μία συνάρτηση η οποία εκτυπώνει τα περιεχόμενα της εικόνας στην οθόνη ως εξής:

- Εκτυπώνει χαρακτήρα αλλαγής γραμμής και το μήνυμα

  "`==========` **IMAGE** `==========`" (10 χαρακτήρες **'='** σε κάθε πλευρά) ακολουθούμενο από χαρακτήρα αλλαγής γραμμής.

- Εκτυπώνει το πλάτος, το ύψος και τη μέγιστη τιμή χρώματος χωρισμένα μεταξύ τους με τον χαρακτήρα κενό (space). Ακολουθεί χαρακτήρας αλλαγής γραμμής.

- Εκτυπώνει ανά γραμμή και ξεκινώντας από την 1η γραμμή, όλες τις τιμές του πίνακα των pixels.. Οι τιμές χωρίζονται μεταξύ τους με τον χαρακτήρα κενό (space). Μετά την εκτύπωση της τελευταίας τιμής κάθε γραμμής ακολουθεί χαρακτήρας αλλαγής γραμμής.

- Εκτυπώνει το μήνυμα "`==========`" (27 χαρακτήρες '=') ακολουθούμενο από δύο χαρακτήρες αλλαγής γραμμής και το διαχωριστικό string.

Έλεγχος: Στη συνάρτηση **main** καλέστε τη συνάρτηση που μόλις γράψατε. Τρέξτε το πρόγραμμά σας προσδιορίζοντας την εικόνα **test.ppm** ώστε να τη φορτώσει και μετά να βεβαιωθείτε ότι η συνάρτησή σας εκτυπώνει σωστά τα περιεχόμενα αυτής της εικόνας. Εφόσον είναι όλα σωστά, βάλτε σε σχόλια τη συγκεκριμένη γραμμή πριν προχωρήσετε στο επόμενο βήμα.

Βήμα 4 – Αποθήκευση της εικόνας σε αρχείο

Στη συνάρτηση **main**, αφού διαβάσετε με επιτυχία την εικόνα, το πρόγραμμα σας θα πρέπει να εκτυπώνει χαρακτήρα αλλαγής γραμμής και το παρακάτω μήνυμα, ακολουθούμενο από χαρακτήρα αλλαγής γραμμής. Το μήνυμα αντιπροσωπεύει το κεντρικό μενού του προγράμματος και εκτυπώνεται αφού το πρόγραμμα διαβάσει την αρχική εικόνα (βήμα 2) και κάθε φορά που ολοκληρώνεται μία εντολή και ο χρήστης καλείται να εισάγει την επόμενη. Κάθε γραμμή μετά το Enter option: ξεκινά με χαρακτήρα tab (`'\t'`).

```
Enter option:
    Print image to screen: (P/p)
    Load another image: (L/l)
    Save image: (S/s)
  Quit: (Q/q)
```

Οι επιλογές οι οποίες δίνονται στον χρήστη είναι οι εξής:

- **`P`** ή **`p`** : εκτυπώνεται η εικόνα στην οθόνη χρησιμοποιώντας τη συνάρτηση εκτύπωσης του βήματος 3.

- **`L`** ή **`l`**: το πρόγραμμα εμφανίζει χαρακτήρα αλλαγής γραμμής και το μήνυμα "**`Enter image path: `**" (με ένα κενό μετά το ':') και περιμένει από το χρήστη να εισάγει ένα file path από το οποίο θα διαβάσει μία νέα εικόνα. Στη συνέχεια, χρησιμοποιεί τη συνάρτηση **`load_image`** για να διαβάσει την εικόνα που βρίσκεται στο file path που έδωσε ο χρήστης. Εάν η συνάρτηση **`load_image`** αποτύχει, εκτυπώνει χαρακτήρα αλλαγής γραμμής και το μήνυμα "**`Invalid path!`**" ακολουθούμενο από χαρακτήρα αλλαγής γραμμής και επαναλαμβάνει τη διαδικασία μέχρι ο χρήστης να δώσει ένα έγκυρο path. Μετά το επιτυχημένο διάβασμα της εικόνας εκτυπώνει χαρακτήρα αλλαγής γραμμής και το μήνυμα "**`Load OK!`**" ακολουθούμενο από το διαχωριστικό string.

- **`S`** ή **`s`**: το πρόγραμμα εμφανίζει χαρακτήρα αλλαγής γραμμής και το μήνυμα "**`Enter new image path: `**" (με ένα κενό μετά το ':') και περιμένει από το χρήστη να εισάγει ένα file path στο οποίο θα αποθηκεύσει την τρέχουσα εικόνα. Στη συνέχεια, χρησιμοποιεί τη συνάρτηση **`save_image`** για να αποθηκεύσει την τρέχουσα εικόνα στο file path που έδωσε ο χρήστης. Η `save_image` λειτουργεί με τρόπο αντίστοιχο της `load_image`. Εάν η **`save_image`** αποτύχει, εκτυπώνει χαρακτήρα αλλαγής γραμμής και το μήνυμα "**`Invalid path!`**" ακολουθούμενο από χαρακτήρα αλλαγής γραμμής κι επαναλαμβάνει τη διαδικασία μέχρι ο χρήστης να δώσει ένα έγκυρο path. Μετά την επιτυχημένη αποθήκευση εκτυπώνει χαρακτήρα αλλαγής γραμμής και το μήνυμα "**`Save OK!`**" ακολουθούμενο από το διαχωριστικό string.

- **`Q`** ή **`q`**: το πρόγραμμα τερματίζει.

Βήμα 5 – Μεταβολή της φωτεινότητας της εικόνας

Μπορείτε να μεταβάλετε τη φωτεινότητα της εικόνας σε πιο ανοιχτή ή πιο σκούρα εάν πολλαπλασιάσετε τις τιμές των τριών χρωμάτων όλων των pixels της εικόνας με έναν αριθμό. Στη συγκεκριμένη άσκηση θα χρησιμοποιήσουμε τον ίδιο αριθμό και για τα τρία χρώματα.

Γράψτε μία συνάρτηση η οποία λαμβάνει ως παραμέτρους ένα πίνακα από struct pixel δύο διαστάσεων, έναν ακέραιο που συμβολίζει το πλάτος της εικόνας, έναν ακέραιο που συμβολίζει το ύψος της εικόνας, έναν ακέραιο που συμβολίζει τη μέγιστη τιμή για όλα τα χρώματα RGB και έναν αριθμό κινητής υποδιαστολής που περιγράφει το συντελεστή μεταβολής της φωτεινότητας με βάση τον οποίο η συνάρτηση θα πολλαπλασιάσει τις τιμές όλων των pixels και για τα τρία RGB χρώματα. Η συνάρτηση δεν επιστρέφει κάτι.

Σημείωση: Εάν η συνάρτηση υπολογίζει τιμές χρώματος μεγαλύτερες της μέγιστης, αυτές θα πρέπει να αποκόπτονται στην τιμή της μέγιστης.

Στη συνάρτηση **main**, συμπληρώστε κώδικα ώστε το κεντρικό μενού του προγράμματος να είναι

```
Enter option:
    Print image to screen: (P/p)
    Load another image: (L/l)
    Save image: (S/s)
    Change luminocity: (U/u)
    Quit: (Q/q)
```

Στη συνάρτηση **main** θα πρέπει να προσθέσετε μία επιπλέον επιλογή προς τον τελικό χρήστη. Εισάγοντας **U** ή **u** το πρόγραμμα εμφανίζει χαρακτήρα αλλαγής γραμμής και το μήνυμα "**Enter luminocity factor:** " (με ένα κενό μετά το ':') και διαβάζει έναν αριθμό κινητής υποδιαστολής που αντιπροσωπεύει το συντελεστή μεταβολής της φωτεινότητας. Εάν δοθεί τιμή εκτός του διαστήματος (0,2] το πρόγραμμα εκτυπώνει χαρακτήρα αλλαγής γραμμής και το μήνυμα "**Factor should be between (0,2]**" ακολουθούμενο από χαρακτήρα αλλαγής γραμμής κι επαναλαμβάνει την ερώτηση προς τον χρήστη μέχρι να δοθεί έγκυρη τιμή.

Εφόσον δοθεί έγκυρη τιμή, το πρόγραμμα σας χρησιμοποιεί τη συνάρτηση που γράψατε σε αυτό το βήμα προκειμένου να μεταβάλει τη φωτεινότητα της εικόνας με βάση τον συντελεστή που εισήγαγε ο χρήστης. Μετά την μετατροπή εκτυπώνει χαρακτήρα αλλαγής γραμμής και το μήνυμα "**Luminocity OK!**" ακολουθούμενο από το διαχωριστικό string.

Βήμα 6 – Μετατροπή της εικόνας σε ασπρόμαυρη

Μια ασπρόμαυρη (*grayscale*) εικόνα είναι μία εικόνα της οποίας και τα τρία RGB χρώματα κάθε pixel έχουν την ίδια τιμή. Για παράδειγμα οι RGB τιμές 0 0 0 αντιπροσωπεύουν το μαύρο χρώμα, οι τιμές 255 255 255 το λευκό και οι τιμές 128 128 128 τη συγκεκριμένη απόχρωση του γκρι.

Γράψτε μία συνάρτηση η οποία λαμβάνει ως παραμέτρους ένα πίνακα από struct pixel δύο διαστάσεων, έναν ακέραιο που συμβολίζει το πλάτος της εικόνας και έναν ακέραιο που συμβολίζει το ύψος της εικόνας. Για κάθε pixel της εικόνας η συνάρτηση υπολογίζει την απόχρωση του γκρι με βάση την παρακάτω φόρμουλα. Η συνάρτηση δεν επιστρέφει κάτι.

```
gray = (red * 0.3 + green * 0.59 + blue * 0.11)
```

Στη συνάρτηση **main**, συμπληρώστε κώδικα ώστε το κεντρικό μενού του προγράμματος να είναι

```
Enter option:
    Print image to screen: (P/p)
    Load another image: (L/l)
    Save image: (S/s)
    Change luminocity: (U/u)
    Convert to grayscale: (G/g)
    Quit: (Q/q)
```

Στη συνάρτηση **main** θα πρέπει να προσθέσετε μία επιπλέον επιλογή προς τον τελικό χρήστη. Εισάγοντας **G** ή **g** το πρόγραμμα καλεί τη συνάρτηση που γράψατε παραπάνω και μετατρέπει την εικόνα σε ασπρόμαυρη. Μετά την μετατροπή εκτυπώνει χαρακτήρα αλλαγής γραμμής και το μήνυμα "**Grayscale OK!**" ακολουθούμενο από το διαχωριστικό string.

Βήμα 7 – Διπλασιασμός του μεγέθους της εικόνας

Γράψτε μία συνάρτηση η οποία λαμβάνει ως παραμέτρους ένα πίνακα από struct pixel δύο διαστάσεων, ένα δείκτη σε ακέραιο που συμβολίζει το πλάτος της εικόνας και ένα δείκτη σε ακέραιο που συμβολίζει το ύψος της εικόνας. Η συνάρτηση εξετάζει εάν η χωρητικότητα του υφιστάμενου πίνακα επαρκεί για τον διπλασιασμό της εικόνας. Εάν όχι, επιστρέφει ένδειξη αποτυχίας. Εάν ναι διπλασιάζει την εικόνα ανανεώνοντας τις διαστάσεις της μέσω των δεικτών προς το πλάτος και ύψος και και επιστρέφει ένδειξη επιτυχίας.

Για να διπλασιαστεί η εικόνα, η τιμή του κάθε pixel στη θέση **`row, col`** αντιγράφεται στις θέσεις

- **`2*row, 2*col,`**
- **`2*row+1, 2*col,`**
- **`2*row, 2*col+1`**
- **`2*row+1, 2*col+1`**

Σημείωση: Ξεκινήστε να κάνετε τις μετατροπές από το τέλος προς την αρχή κάθε γραμμής και κάθε στήλης. **ΔΕΝ επιτρέπεται να χρησιμοποιήσετε βοηθητικό πίνακα.**

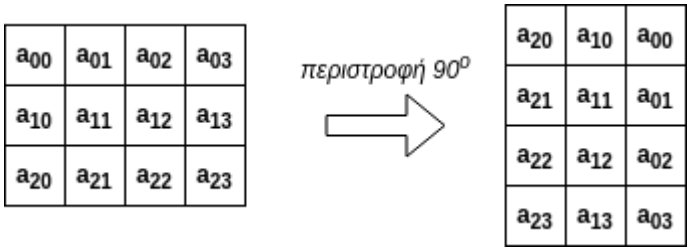
Στη συνάρτηση **main**, συμπληρώστε κώδικα ώστε το κεντρικό μενού του προγράμματος να είναι

```
Enter option:
    Print image to screen: (P/p)
    Load another image: (L/l)
    Save image: (S/s)
    Change luminocity: (U/u)
    Convert to grayscale: (G/g)
    Duplicate size: (D/d)
    Quit: (Q/q)
```

Στη συνάρτηση **main** θα πρέπει να προσθέσετε μία επιπλέον επιλογή προς τον τελικό χρήστη. Εισάγοντας **D** ή **d** το πρόγραμμα καλεί τη συνάρτηση που γράψατε παραπάνω. Μετά την μετατροπή εκτυπώνει χαρακτήρα αλλαγής γραμμής και το μήνυμα "**`Duplication OK!`**" ή "**`Duplication NOT OK!`**" ανάλογα με το αν ο διπλασιασμός έγινε επιτυχώς ή όχι, ακολουθούμενο από το διαχωριστικό string.

Βήμα 8 – Δεξιά περιστροφή της εικόνας κατά 90°

Γράψτε μία συνάρτηση η οποία λαμβάνει ως παραμέτρους ένα πίνακα από struct pixel δύο διαστάσεων, ένα δείκτη σε ακέραιο που συμβολίζει το πλάτος της εικόνας και ένα δείκτη σε ακέραιο που συμβολίζει το ύψος της εικόνας και δεν επιστρέφει κάτι. Η συνάρτηση περιστρέφει δεξιόστροφα την εικόνα και ανανεώνοντας τις διαστάσεις της μέσω των δεικτών προς το πλάτος και ύψος. Μπορείτε να εξάγετε τον αλγόριθμο περιστροφής από την μεταβολή στη θέση των στοιχείων της διπλανής εικόνας. Μπορείτε να χρησιμοποιήσετε βοηθητικό πίνακα μέσα στη συνάρτηση που θα γράψετε.



Στη συνάρτηση **main**, συμπληρώστε κώδικα ώστε το κεντρικό μενού του προγράμματος να είναι

```
Enter option:
   Print image to screen: (P/p)
    Load another image: (L/l)
    Save image: (S/s)
    Change luminocity: (U/u)
    Convert to grayscale: (G/g)
    Duplicate size: (D/d)
    Rotation: (R/r)
    Quit: (Q/q)
```

Στη συνάρτηση **main** θα πρέπει να προσθέσετε μία επιπλέον επιλογή προς τον τελικό χρήστη. Εισάγοντας **R** ή **r** το πρόγραμμα καλεί τη συνάρτηση που γράψατε παραπάνω και περιστρέφει την εικόνα κατά 90°. Μετά την μετατροπή εκτυπώνει χαρακτήρα αλλαγής γραμμής και το μήνυμα "**`Rotation OK!`**" ακολουθούμενο από το διαχωριστικό string.

Άσκηση 2: Έλεγχος ομοιότητας

Στην παρούσα άσκηση θα γράψετε ένα πρόγραμμα το οποίο ελέγχει το βαθμό ομοιότητας δύο συμβολοσειρών που αναπαριστούν πραγματική (του φοιτητή) και αναμενόμενη (του καθηγητή) έξοδο κάποιου προγράμματος και παράγει αντιπροσωπευτικό βαθμό (ένα πολύ απλό δικό σας autolab).

Σας δίνουμε το αρχείο `hw3b.c` το οποίο περιέχει ήδη κάποιους ορισμούς. Προσθέστε τον κώδικά σας σε αυτό.

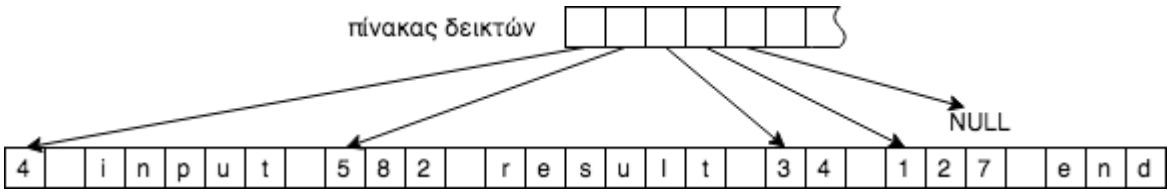
Μορφή δεδομένων και έλεγχος ομοιότητας.

Υποθέτουμε ότι κάθε μια συμβολοσειρά προς έλεγχο αποτελείται από μια σειρά από "λέξεις" (tokens) με κενό ανάμεσα σε διαδοχικά tokens. Κάθε ένα token μπορεί να αποτελείται είτε αποκλειστικά από μικρά γράμματα είτε αποκλειστικά από ψηφία. Για παράδειγμα, η αναμενόμενη έξοδος θα μπορούσε να είναι:

```
"4 input 582 result 34 127 end"
```

Επειδή τα tokens ψηφίων που εμφανίζονται στις συμβολοσειρές αντιστοιχούν σε αποτελέσματα (π.χ. 4, 582, 34, 127) ενώ τα tokens γραμμάτων (π.χ. input, result, end) είναι απλά μηνύματα, θέλουμε να ελέγξουμε ξεχωριστά τις ομοιότητες ανάμεσα σε tokens ψηφίων από τις ομοιότητες ανάμεσα σε tokens γραμμάτων και να δώσουμε διαφορετικό βάρος σε αυτές. Ο έλεγχος για τα tokens ψηφίων θα γίνει ως εξής:

Θα κατασκευάσουμε ένα πίνακα δεικτών σε χαρακτήρα έτσι ώστε κάθε κελί του να δείχνει στην αρχή καθενός token ψηφίων. Αυτό θα το κάνουμε και για την αναμενόμενη και για την πραγματική έξοδο. Για το παράδειγμα που είδατε πιο πάνω, ο πίνακας δεικτών θα είναι όπως στο παρακάτω σχήμα:



Εφόσον κάνουμε το ίδιο και για τις δύο συμβολοσειρές, μπορούμε να συγκρίνουμε ένα προς ένα τα tokens που μας ενδιαφέρουν. Για παράδειγμα, αν το παραπάνω αντιστοιχεί στην αναμενόμενη έξοδο ενώ η πραγματική είναι `"8 582 input result 4 128 and 45"` τότε είναι σωστό μόνο ένα από τα τέσσερα tokens ψηφίων (το 582). Παρατηρήστε ότι παρόλο που εμφανίζεται το 4 στην πραγματική έξοδο, δε θα το θεωρήσουμε σωστό διότι δεν είναι στη σωστή θέση σε σχέση με τα υπόλοιπα tokens ψηφίων. Ουσιαστικά συγκρίνουμε το 4 με το 8, το 582 με το 582, το 34 με το 4 και το 127 με το 128 και σταματάμε γιατί η αναμενόμενη έξοδος δεν έχει άλλα tokens ψηφίων.

Αν θεωρήσουμε ότι θέλουμε να αναθέσουμε συνολικά 100 πόντους, εκ των οποίων 80 για την ομοιότητα σε tokens ψηφίων και 20 για την ομοιότητα σε tokens γραμμάτων, τότε το σκορ για τα tokens ψηφίων θα είναι 20 (το 1/4 του 80 εφόσον είναι σωστό ένα από τα 4 tokens).

Αντίστοιχα, κάνουμε τον έλεγχο και υπολογίζουμε το σκορ για τα tokens γραμμάτων. Στο συγκεκριμένο παράδειγμα, αυτό θα είναι 13.33 γιατί είναι σωστά δύο από τα τρία tokens γραμμάτων.

Βήμα 1 – Δηλώσεις και εισαγωγή δεδομένων

Δηλώστε δύο πίνακες χαρακτήρων μεγέθους MAX_SIZE έναν στον οποίο θα αποθηκευτεί η πραγματική έξοδος ενός προγράμματος κι έναν στον οποίο θα αποθηκευτεί η αναμενόμενη έξοδος.

Εκτυπώστε το μήνυμα "**`Actual:`** " με ένα κενό μετά το ':', διαβάστε την πραγματική έξοδο ενός προγράμματος και αποθηκεύστε την στον αντίστοιχο πίνακα. Συνίσταται η χρήση `fgets`. Λάβετε υπόψη ότι η **`fgets`** αποθηκεύει στη συμβολοσειρά που διάβασε και το '\n' που πληκτρολόγησε ο χρήστης. Δείτε περισσότερα στο παράρτημα στο τέλος της εκφώνησης.

Εκτυπώστε το μήνυμα "**`Expected:`** " με ένα κενό μετά το ':', διαβάστε την αναμενόμενη έξοδο ενός προγράμματος και αποθηκεύστε την στον αντίστοιχο πίνακα.

Βήμα 2 – Έλεγχοι

(α)

Γράψτε μια συνάρτηση σκοπός της οποίας είναι να κατασκευάσει ένα πίνακα δεικτών ώστε αυτοί να δείχνουν στα κατάλληλα tokens του πίνακα χαρακτήρων.

Η συνάρτηση παίρνει ως παραμέτρους έναν αρχικά άδειο πίνακα δεικτών σε χαρακτήρα, ένα πίνακα χαρακτήρων και ένα πίνακα που περιέχει όλους τους χαρακτήρες που θέλουμε να υπάρχουν στα tokens που μας ενδιαφέρουν (για παράδειγμα, η τρίτη παράμετρος θα είναι "0123456789" αν θέλουμε να χρησιμοποιήσουμε τη συνάρτηση για να βρούμε tokens ψηφίων και "abcdefghijklmnopqrstuvwxyz" αν θέλουμε να χρησιμοποιήσουμε τη συνάρτηση για να βρούμε tokens γραμμάτων).

Στο τέλος της συνάρτησης, ο πίνακας δεικτών θα πρέπει να έχει αρχικοποιηθεί ώστε κάθε κελί του να δείχνει στην αρχή καθενός token που μας ενδιαφέρει. Κελιά που δε χρησιμοποιούνται θα πρέπει να περιέχουν NULL.

Βεβαιωθείτε ότι η συνάρτησή σας λειτουργεί σωστά πριν συνεχίσετε.

(β)

Γράψτε μια συνάρτηση η οποία παίρνει ως παραμέτρους δύο πίνακες χαρακτήρων που θα περιέχουν την αναμενόμενη και την πραγματική έξοδο αντίστοιχα, ένα πίνακα που περιέχει όλους τους χαρακτήρες που θέλουμε να υπάρχουν στα tokens που μας ενδιαφέρουν, κι έναν ακέραιο που αναπαριστά το μέγιστο πλήθος πόντων που αντιστοιχούν στον έλεγχο που κάνουμε.

Η συνάρτηση καλεί δύο φορές τη συνάρτηση που γράψατε στο βήμα 2(α), μία για να κατασκευάσει τον πίνακα δεικτών για την αναμενόμενη έξοδο, και μία για την πραγματική έξοδο. Ακολούθως, συγκρίνει ένα-προς-ένα τα αντίστοιχα tokens και υπολογίζει πόσα από αυτά είναι σωστά.

Εκτυπώνει χαρακτήρα αλλαγής γραμμής, το μήνυμα "**`Correct tokens = X out of Y`**" και χαρακτήρα αλλαγής γραμμής. X είναι το πλήθος σωστών tokens και Y το συνολικό πλήθος tokens (του τύπου που ελέγχουμε) που περιλαμβάνει η αναμενόμενη έξοδος. Για το παράδειγμα του βήματος 2, η συνάρτηση θα πρέπει να εκτυπώσει Correct tokens = 1 out of 4 όταν την καλέσουμε να ελέγξει τα tokens ψηφίων και Correct tokens = 2 out of 3 όταν την καλέσουμε να ελέγξει τα tokens γραμμάτων.

Τέλος, η συνάρτηση υπολογίζει κι επιστρέφει το πλήθος πόντων που θα πάρει ο φοιτητής με βάση το πόσα σωστά tokens έχουν βρεθεί και ποιο είναι το μέγιστο πλήθος πόντων.

Βεβαιωθείτε ότι η συνάρτησή σας λειτουργεί σωστά πριν συνεχίσετε.

(γ)

Στη main καλέστε τη συνάρτηση που γράψατε στο βήμα 2(β) δύο φορές, πρώτα μία για να υπολογίσετε τους πόντους από την ομοιότητα των tokens ψηφίων για την αναμενόμενη και πραγματική έξοδο που διαβάσατε, και μετά μία για να υπολογίσετε τους πόντους από την ομοιότητα των tokens γραμμάτων γι αυτές τις εξόδους.

Τελικά εκτυπώστε χαρακτήρα αλλαγής γραμμής, το μήνυμα "**`T.TT (D.DD+L.LL)`**" και χαρακτήρα αλλαγής γραμμής. `T.TT` είναι οι συνολικοί πόντοι που παίρνει ο φοιτητής, `D.DD` είναι οι πόντοι που πήρε για την ομοιότητα των tokens ψηφίων και `L.LL` οι πόντοι για την ομοιότητα των tokens γραμμάτων. Όλοι οι αριθμοί εμφανίζονται με δύο δεκαδικά ψηφία.

BONUS – Αποκλειστική χρήση string.h

Για επιπλέον βαθμολογικό bonus, γράψτε τη λύση σας ώστε να χρησιμοποιεί αποκλειστικά συναρτήσεις από το string.h, χωρίς καμία επανάληψη που να διατρέχει πίνακα ένα-ένα χαρακτήρα.

Στην τελευταία σελίδα θα βρείτε πληροφορίες για δύο συναρτήσεις που θα σας φανούν χρήσιμες.

Πώς να παραδώσετε τη δουλειά σας

Πριν παραδώσετε το πρόγραμμά σας, προσθέστε σε σχόλια στην αρχή του αρχείου τα πλήρη ονόματα και ΑΕΜ των μελών της ομάδας. Παρακαλούμε να γράφετε τα σχόλια ΜΟΝΟ με λατινικούς χαρακτήρες.

1. **Κατασκευάστε ένα κατάλογο** με όνομα `hw3submit` και αντιγράψτε μέσα σε αυτόν το `hw3a.c` και το `hw3b.c`
2. Κατασκευάστε ένα αρχείο με όνομα team.txt και προσθέστε σε αυτό τα πλήρη ονόματα και ΑΕΜ των μελών της ομάδας, ακόμη κι αν η ομάδα αποτελείται από ένα άτομο.
3. **Κάντε δεξί κλικ** στον κατάλογο `hw3submit` και επιλέξτε `Compress here as tar.gz`
4. **Πηγαίνετε στο Autolab**,
5. **Επιλέξτε το hw3.**
   (a) Αν είστε ομάδα δύο ατόμων, **κατασκευάστε** μια ομάδα μέσω της επιλογής Group Options. Αυτή θα ισχύει μόνο για το hw3.
   (b) Κάντε Submit το hw3submit.tar.gz στο Autolab.

ΠΑΡΑΡΤΗΜΑ

`fgets`

Σε αυτή την εργασία σας προτείνουμε να χρησιμοποιήσετε τη συνάρτηση `fgets` για να διαβάσετε μία-μία γραμμή δεδομένων από το πληκτρολόγιο. Η `fgets` παίρνει τρεις παραμέτρους:

- Τη διεύθυνση στην οποία θέλουμε να αποθηκευτεί η γραμμή που διαβάζεται
- Το μέγιστο πλήθος bytes που θέλουμε να διαβαστούν συν ένα.
- Από πού θέλουμε να διαβαστούν τα δεδομένα. Όταν θέλουμε να διαβάσουμε από το πληκτρολόγιο, προσδιορίζουμε ως παράμετρο το `stdin`.

Για παράδειγμα, αν υποθέσουμε ότι έχουμε τον πίνακα `char str[SIZE]`, τότε μπορούμε να διαβάσουμε μια γραμμή από το πληκτρολόγιο και να την αποθηκεύσουμε στον πίνακα γράφοντας

```
fgets(str, SIZE, stdin)
```

Η `fgets` θα αρχίσει να διαβάζει μέχρι είτε να συναντήσει χαρακτήρα αλλαγής γραμμής, είτε να έχει διαβάσει μέχρι και `SIZE-1` χαρακτήρες, όποιο συμβεί πρώτο. Στους χαρακτήρες που διάβασε θα προσθέσει στο τέλος και το '\0' (γι αυτό διάβασε ένα λιγότερο από όσο προσδιορίσαμε στη δεύτερη παράμετρο) και θα τους αποθηκεύσει όλους στο `str`.

Μια ιδιαιτερότητα της `fgets` είναι ότι αποθηκεύει στον πίνακα και το χαρακτήρα αλλαγής γραμμής εφόσον τον διαβάσει.

Παραδείγματα για `SIZE` ίσο με `5`:

- Αν στο πληκτρολόγιο γράψουμε bye τότε θα διαβαστούν 4 χαρακτήρες (τα 'b', 'y', 'e', '\n') και στον πίνακα `str` θα αποθηκευτούν αυτοί καθώς και το '\0' στο τέλος.
- Αν στο πληκτρολόγιο γράψουμε `hello` και πατήσουμε enter τότε θα διαβαστούν `SIZE-1` δηλαδή 4 χαρακτήρες και στον πίνακα θα αποθηκευτεί η συμβολοσειρά `"hell"`. Αν αμέσως μετά ξανακαλέσουμε την `fgets` με τον ίδιο τρόπο, τότε θα διαβαστούν και αυτοί που απομένουν, δηλαδή το 'o' και το '\n'.

Αν δεν επιθυμούμε να υπάρχει '\n' στο τέλος, πρέπει πρώτα να ελέγξουμε αν πράγματι έχει αποθηκευτεί στον πίνακα και, αν ναι, να το αντικαταστήσουμε με '\0'.

`strspn`

Αυτή η συνάρτηση χρησιμοποιείται όταν θέλουμε να βρούμε μέχρι ποιο σημείο μιας συμβολοσειράς εμφανίζονται συγκεκριμένοι χαρακτήρες. Η `strspn` παίρνει ως παραμέτρους δύο συμβολοσειρές, ας πούμε `str` και και `set` και βρίσκει κι επιστρέφει τη θέση του πρώτου χαρακτήρα στο `str` ο οποίος δεν εμφανίζεται στο `set`. Το όνομα `strspn` προέρχεται από τη φράση string span.

Παραδείγματα:

- `strspn("3210boom", "012345")` επιστρέφει `4` δηλαδή τη θέση του `'b'` εφόσον το `'b'` είναι ο πρώτος χαρακτήρας του `"3210boom"` που δεν εμφανίζεται στο `"012345"`
- `strspn("3210boom", "d2m x\n")` επιστρέφει `0` δηλαδή τη θέση του `'3'` διότι αυτό είναι ο πρώτος χαρακτήρας του `"3210boom"` που δεν εμφανίζεται στο `"d2m x\n"`. Παρατηρήστε πως το `'m'` εμφανίζεται στο `"d2m x\n"` αλλά δε θα φτάσει ποτέ ως εκεί ο έλεγχος γιατί γίνεται από την αρχή του `"3210boom"` και σταματά πάντα στον πρώτο χαρακτήρα που αποτυγχάνει.
- `strspn("3210boom", "book123m0")` επιστρέφει `8` δηλαδή τη θέση του `'\0'` στο `"3210boom"` διότι όλοι οι χαρακτήρες του `"3210boom"` εμφανίζονται στο δεύτερο `"d2m x\n"` και ο έλεγχος σταματά στο σημείο που το `"3210boom"` τερματίζει.

`strpbrk`

Αυτή η συνάρτηση χρησιμοποιείται όταν θέλουμε να βρούμε το πρώτο σημείο μιας συμβολοσειράς στο οποίο εμφανίζεται ένας χαρακτήρας από κάποιο σύνολο. Η `strpbrk` παίρνει ως παραμέτρους δύο συμβολοσειρές, ας πούμε `str` και και `set` και βρίσκει κι επιστρέφει τη διεύθυνση του πρώτου χαρακτήρα στο `str` ο οποίος εμφανίζεται στο `set`. Αν δεν υπάρχει κανένας, επιστρέφει NULL. Το όνομα `strpbrk` προέρχεται από τη φράση string pointer break.

Παραδείγματα:

- `strpbrk("3210boom", "abcdef")` επιστρέφει τη διεύθυνση του `'b'` (δηλαδή δείκτη στο `'b'`) εφόσον το `'b'` είναι ο πρώτος χαρακτήρας του `"3210boom"` που εμφανίζεται στο `"abcdef"`
- `strpbrk("3210boom", "d2\n")` επιστρέφει `NULL` δηλαδή τη θέση του `'3'` διότι κανένας χαρακτήρας του `"3210boom"` δεν εμφανίζεται στο `"d2\n"`. Παρατηρήστε πως ένας από τους χαρακτήρες του `"d2\n"` είναι ο χαρακτήρας αλλαγής γραμμής.