

Προγραμματισμός II
3^ο project

Διδάσκων:
Χ. Τρυφωνόπουλος

Παράδοση μέχρι: Κυριακή 18/07/2021 ώρα 23.59

ΣΗΜΑΝΤΙΚΕΣ ΣΗΜΕΙΩΣΕΙΣ:

1. Στο αρχείο που γράφετε τον κώδικα για κάθε εργασία πρέπει ΟΠΩΣΔΗΠΟΤΕ να βάλετε σε σχόλια τα ονόματα, τους A.M., και τα username/email των μελών της ομάδας (ομάδες αυστηρά 2 ατόμων, ατομικές εργασίες δεν γίνονται δεκτές). Όλα τα σχόλια και τα μηνύματα του προγράμματός σας πρέπει να είναι με λατινικούς χαρακτήρες (και όχι στα ελληνικά). Ακολουθεί παράδειγμα:
/* Nikos Papadopoulos
AM: 2022201500666
dit15666@uop.gr

Christos Tryfonopoulos
AM: 2022201600777
dit16777@uop.gr
*/
2. Αφού έχετε ολοκληρώσει την εργασία που θέλετε να παραδώσετε την υποβάλετε στο eclass στο υποσύστημα «Εργασίες φοιτητών». Προσοχή: μόνο 1 άτομο από την ομάδα χρειάζεται να παραδώσει την εργασία μέσω του e-class! Η υποβολή πρέπει να γίνει ΠΡΙΝ την ημερομηνία παράδοσης. Παραδίδετε ΜΟΝΟ τα αρχεία με τον κώδικα (με κατάληξη .c ή/και .h) σε ένα συμπίεσμένο αρχείο (το οποίο θα φέρει τα ονόματα της ομάδας π.χ., PapadopoulosTryfonopoulos.zip) και ΟΧΙ τα εκτελέσιμα μετά την μεταγλώττιση. Προσοχή: τα προγράμματα που θα παραδώσετε θα πρέπει να κάνουν compile και να τρέχουν στο pelopas.uop.gr (2^ο έτος ή μεγαλύτερο) ή στο hilon.dit.uop.gr (1^ο έτος). Ασκήσεις οι οποίες δεν κάνουν compile ή δεν τρέχουν στα παραπάνω μηχανήματα του τμήματος θα μηδενίζονται.
3. Περιπτώσεις αντιγραφής θα μηδενίζονται μαζί με όλες τις ασκήσεις που έχουν ήδη παραδώσει και οι εμπλεκόμενοι δεν θα έχουν δικαίωμα παράδοσης άλλων ασκήσεων. Επιπλέον θα παραπέμπονται για περαιτέρω κυρώσεις στα αρμόδια όργανα του Τμήματος.
4. Η ημερομηνία παράδοσης είναι αυστηρή, και η παράδοση γίνεται μόνο μέσω του eclass και όχι με email στον διδάσκοντα ή στους βοηθούς του μαθήματος. Ασκήσεις που παραδίδονται μετά τη λήξη της προθεσμίας δε γίνονται δεκτές.
5. Καθώς η άσκηση δεν έχει προσωπική εξέταση για όλους τους συμμετέχοντες, δικαίωμα να παραδώσουν την άσκηση έχουν όσοι είχαν βαθμό ≥ 4 σε μία τουλάχιστον από τις προηγούμενες ασκήσεις. Για όσες ομάδες κρίνεται απαραίτητο θα γίνει προσωπική εξέταση το Σεπτέμβριο.

ΕΚΦΩΝΗΣΗ ΕΡΓΑΣΙΑΣ

Στην εργασία αυτή θα πρέπει να υλοποιήσετε έναν απλό επεξεργαστή εικόνας που θα ονομάζεται BiP (Bitmap Processor) ο οποίος θα μπορεί να διαβάσει, να επεξεργαστεί, και να αποθηκεύσει εικόνες που είναι σε bitmap format (.bmp). Το format αυτό είναι γενικά απλό στην επεξεργασία του και είναι εύκολο να γράψει κανείς προγράμματα που να το διαχειρίζονται. Έχει πολλές παραλλαγές, αλλά εμείς θα ασχοληθούμε μόνο με αρχεία bmp που έχουν 24-bit χρώμα και δεν χρησιμοποιούν συμπίεση καθώς είναι τα απλούστερα.

1. Επεξήγηση του bitmap format

Στα αρχεία που είναι σε bitmap format, ένα pixel, δηλαδή μια κουκκίδα στην εικόνα, αντιστοιχεί σε 3 bytes, ένα για κάθε βασικό χρώμα: κόκκινο, πράσινο, και μπλέ (Red, Green, Blue – RGB). Αυτά τα bytes τα χειριζόμαστε σαν unsigned char, άρα περιέχουν τιμές στο διάστημα από 0 έως 255. Όσο πιο μικρή είναι η τιμή του byte τόσο λιγότερο από αυτό το χρώμα υπάρχει στο pixel. Άρα το (0,0,0) σημαίνει απουσία οποιουδήποτε χρώματος και αντιστοιχεί στο μαύρο, ενώ το (255,255,255) έχει την μέγιστη ποσότητα από όλα τα χρώματα και αντιστοιχεί στο άσπρο. Προφανώς το (255,0,0) είναι το κόκκινο, το (0,255,0) είναι το πράσινο, και το (0,0,255) είναι το μπλε.

Μια εικόνα bmp είναι ουσιαστικά ένας δισδιάστατος πίνακας από pixels, τα οποία αποθηκεύονται σε ένα δυαδικό αρχείο ανά γραμμή. Αν μια γραμμή της εικόνας έχει 512 pixels τότε αυτά θα αποθηκευτούν στο αρχείο σαν μια σειρά από τριάδες bytes ξεκινώντας από το αριστερότερο pixel της εικόνας και καταλήγοντας στο δεξιότερο. **Προσοχή** όμως σε τρία πράγματα:

- Κάθε γραμμή από bytes στο αρχείο πρέπει να είναι πολλαπλάσιο του 4. Αν η γραμμή περιέχει λιγότερα bytes (δηλαδή ο αριθμός των bytes δεν είναι πολλαπλάσιο του 4) τότε συμπληρώνουμε τη γραμμή με όσα bytes χρειάζεται (που έχουν την τιμή 0) μέχρι να φτάσουμε στο επόμενο πολλαπλάσιο του 4. Για παράδειγμα, αν η εικόνα έχει πλάτος 53 pixels, τότε με 3 bytes/pixel θα είχαμε στο αρχείο $53 \times 3 = 159$ bytes ανά γραμμή. Το πρώτο πολλαπλάσιο του 4 που είναι μεγαλύτερο του 159 είναι το 160, άρα θα συμπληρώσουμε με 1 ακόμα byte κάθε γραμμή (αφού όλες οι γραμμές θα έχουν μέγεθος 53 pixels). Το byte αυτό θα έχει την τιμή 0 όπως είπαμε και θα βρίσκεται στο τέλος κάθε γραμμής.
- Οι γραμμές αποθηκεύονται με αντίστροφη σειρά, δηλαδή αντί να ξεκινάμε από την πρώτη γραμμή pixels της εικόνας, ξεκινάμε από την τελευταία.
- Ξεκινώντας από αριστερά προς τα δεξιά σε μια γραμμή τα χρώματα είναι αποθηκευμένα με την εξής σειρά: (blue, green, red). Προσέξτε πως είναι ανάποδα από το RGB που έχουμε συνηθίσει (δηλαδή το 1^ο byte είναι το B, το 2^ο byte το G, και το 3^ο byte το R για κάθε pixel).

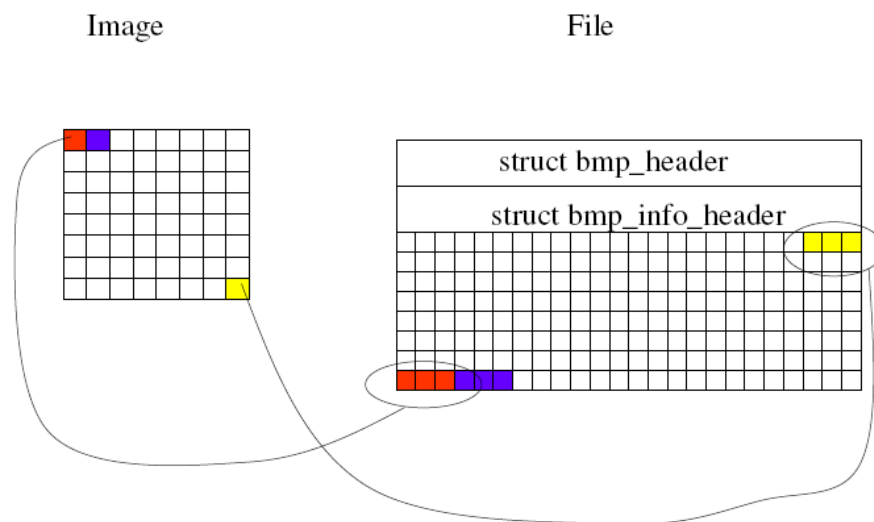
Το αρχείο, εκτός από το χρώμα του κάθε pixel, περιέχει κι άλλες πληροφορίες, όπως το μέγεθος του αρχείου, το πλάτος και το μήκος της εικόνας σε pixels, κ.ά. Αυτές οι πληροφορίες περιέχονται στα ακόλουθα 2 structs:

```

/**** BMP file header structure ****/
struct bmp_header
{
    unsigned short bfType; /* Magic number for file */
    unsigned int bfSize; /* Size of file */
    unsigned short bfReserved1; /* Reserved */
    unsigned short bfReserved2; /* Also reserved */
    unsigned int bfOffBits; /* Offset to bitmap data */
};

/**** BMP file info structure ****/
struct bmp_info_header {
    unsigned int biSize; /* Size of info header */
    int biWidth; /* Width of image */
    int biHeight; /* Height of image */
    unsigned short biPlanes; /* Number of colour planes */
    unsigned short biBitCount; /* Number of bits per pixel */
    unsigned int biCompression; /* Type of compression to use */
    unsigned int biSizeImage; /* Size of image data */
    int biXPelsPerMeter; /* X pixels per meter */
    int biYPelsPerMeter; /* Y pixels per meter */
    unsigned int biClrUsed; /* Number of colours used */
    unsigned int biClrImportant; /* Number of important colours */
};

```



Το αρχείο περιέχει στην αρχή το struct bmp_header και μετά το bmp_info_header. Για να γίνει πιο κατανοητή η δομή του bmp αρχείου δείτε και στο παραπάνω σχήμα όπου φαίνονται όλες οι σχετικές επεξηγήσεις.

Από τις πληροφορίες που βρίσκονται στα structs, εμείς ουσιαστικά χρειαζόμαστε τα biWidth και biHeight, που είναι οι διαστάσεις της εικόνας σε pixels (προσοχή όχι σε bytes!!!). Τα υπόλοιπα δεν θα τα χρησιμοποιήσουμε παρά μόνο για να τυπώσουμε πληροφορίες για το αρχείο.

2. Βασικές λειτουργίες του προγράμματος

Στην ενότητα αυτή περιγράφεται η βασική λειτουργικότητα του προγράμματος που θα πρέπει να υλοποιησετε.

2.1. Χειρισμός αποκλειστικά από τη γραμμή εντολών [10 μονάδες]

Το πρόγραμμα θα το χειριζόμαστε αποκλειστικά από τη γραμμή εντολών και δεν θα έχει καθόλου μενού. Όλη η λειτουργικότητα θα καλείται με κατάλληλα ορίσματα γραμμής εντολών που θα είναι της μορφής:

```
$> ./bip <-option> [input_file] [output_file]
```

όπου:

- το όρισμα option σχετίζεται με επιμέρους λειτουργίες που περιγράφονται παρακάτω (εκτύπωση χαρακτηριστικών, καθρέφτισμα, καλλιτεχνικό ασπρόμαυρο, παλέτα χρωμάτων) καθώς και με διάφορες παραμέτρους που παίρνει κάθε λειτουργία option και περιγράφονται στις επόμενες ενότητες,
- το όρισμα input_file είναι το όνομα της εικόνας προς επεξεργασία χωρίς την προέκταση (η οποία είναι πάντα bmp), και
- το όρισμα output_file είναι το όνομα της επεξεργασμένης εικόνας χωρίς την προέκταση (η οποία είναι πάντα bmp).

Τα αρχεία θα ανοίγονται ως δυαδικά (όχι ως κειμένου, γιατί δεν θα μπορείτε να διαβάσετε τα περιεχόμενά τους) και θα διαβάζετε τα περιεχόμενα των δύο structs που αναφέραμε παραπάνω χρησιμοποιώντας κώδικα που περιέχεται στα αρχεία bmp.c και bmp.h. Συγκεκριμένα, οι συναρτήσεις:

```
readHeader(FILE *fp, struct bmp_header *header)
```

και

```
readInfo(FILE *fp, struct bmp_info_header *header)
```

διαβάζουν το αντίστοιχο struct από το αρχείο και το αποθηκεύουν στη μνήμη στην περιοχή που δείχνει η δεύτερη παράμετρος κάθε συνάρτησης. Προσοχή! Η μνήμη αυτή θα πρέπει να δεσμευτεί από εσάς με χρήση της συνάρτησης malloc!

Αφού διαβάσετε τα structs θα πρέπει να διαβάσετε με την fread την εικόνα εισόδου γραμμή-γραμμή και να την αποθηκεύσετε σε έναν προσωρινό χώρο που θα έχετε δεσμεύσει δυναμικά, ώστε να κάνετε την επεξεργασία που περιγράφεται παρακάτω. Ο χώρος αυτός θα είναι ένας πίνακας από unsigned char με τόσες γραμμές όσες και ο αριθμός των γραμμών της εικόνας (βρίσκεται στο biHeight),

ενώ ο αριθμός των στηλών θα είναι όσες και οι στήλες της εικόνας (ο αριθμός που είναι αποθηκευμένος στο `biWidth`) τροποποιημένος κατάλληλα ώστε να διαιρείται με το 4, με βάση τα όσα είπαμε παραπάνω.

Για να γράψετε την εικόνα που προκύπτει από το αποτέλεσμα της επεξεργασίας στο αρχείο εξόδου (το όνομά τους περιγράφηκε παραπάνω), θα γράψετε στην αρχή του αρχείου εξόδου τα structs, χρησιμοποιώντας τις συναρτήσεις:

```
writeHeader(FILE *fp, struct bmp_header *header)
```

και

```
writelnInfo(FILE *fp, struct bmp_info_header *header)
```

και μετά την πληροφορία για τα pixels της εικόνας με τη συνάρτηση `fwrite`. Μην ξεχάσετε να αποδεσμεύσετε τη μνήμη όπου προσωρινά είχατε αποθηκεύσει την εικόνα καθώς και όποια άλλη μνήμη είχατε δεσμεύσει με `malloc` και να κλείσετε τα αρχεία των εικόνων.

Στη συνέχεια περιγράφονται οι επιλογές επεξεργασίας που πρέπει να υλοποιήσετε καθώς και ο τρόπος κλήσης τους από τη γραμμή εντολών. Προσέξτε ότι το πρόγραμμά σας δεν θα πρέπει να τροποποιεί την αρχική εικόνα (input) σε καμία από τις εντολές επεξεργασίας. Το αποτέλεσμα θα πρέπει να αποθηκεύεται σε νέο αρχείο (οπότε θα πρέπει να ελέγχετε ότι το αρχείο εξόδου δεν είναι το ίδιο με το αρχείο εισόδου).

2.2. Εκτύπωση των χαρακτηριστικών της εικόνας [5 μονάδες]

Η λειτουργία `-a` (από το `attributes`) δεν παίρνει παραμέτρους και χρησιμοποιείται για να τυπώσει τα περιεχόμενα των δύο structs που αναφέραμε παραπάνω (`struct bmp_header` και `struct bmp_info_header`). Επίσης καθώς η λειτουργία αυτή είναι πληροφοριακή και δεν κάνει κάποια επεξεργασία στην εικόνα δεν χρειάζεται να δοθεί όνομα για το `output`. Για παράδειγμα η κλήση:

```
$> ./bip -a balloons
```

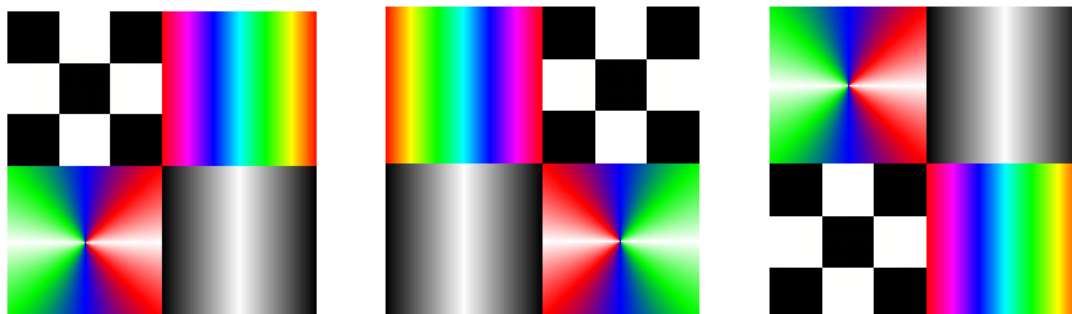
τυπώνει στην οθόνη όλα τα πεδία των δυο παραπάνω structs για την εικόνα `balloons.bmp` (δηλαδή τα `bfType`, `bfSize`, `bfReserved1`, `bfReserved2`, και `bfOffBits` του `bmp_header` και τα `biSize`, `biWidth`, `biHeight`, `biPlanes`, `biBitCount`, `biCompression`, `biSizeImage`, `biXPelsPerMeter`, `biYPelsPerMeter`, `biClrUsed`, και `biClrImportant` του `bmp_info_header`).

2.3. Καθρέφτισμα της εικόνας [20 μονάδες]

Η λειτουργία `-fh` ή `-fv` (από το `flip horizontally` ή `flip vertically`) καθρεφτίζει την εικόνα με βάση τον άξονα στο οποίο θα γίνει το καθρέφτισμα. Για παράδειγμα η κλήση:

```
$> ./bip -fv books books_mirroredV
```

καθρεφτίζει την εικόνα books.bmp κατά τον κατακόρυφο άξονα. Στις παρακάτω εικόνες βλέπουμε το αποτέλεσμα της επεξεργασίας της αρχικής εικόνας (αριστερά), όταν την καθρεφτίσουμε κατά τον οριζόντιο άξονα (μέση) ή κατά τον κατακόρυφο άξονα (δεξιά).





2.4. Μετατροπή σε καλλιτεχνικό ασπρόμαυρο [30 μονάδες]

Στη λειτουργία αυτή ο χρήστης θα δίνει ένα όνομα χρώματος και όλη η εικόνα θα γίνεται ασπρόμαυρη εκτός από το Χ% (ένα μεταβλητό ποσοστό) των pixel των οποίων το χρώμα είναι εγγύτερα στο δοσμένο από το χρήστη χρώμα.

Για να κάνετε μέρος της εικόνας ασπρόμαυρο θα πρέπει να μετατρέψετε κάθε pixel σε μία απόχρωση του γκρι. Προσέξτε ότι εάν ένα pixel έχει την ίδια ποσότητα και από τα 3 χρώματα (RGB), τότε φαίνεται σαν απόχρωση του γκρι (παρατηρήστε ότι το άσπρο είναι (255,255,255), το μαύρο (0,0,0), και το γκρι (127,127,127)). Έτσι, αν έχω το pixel (120,70,96) πρέπει να το μετατρέψω σε ένα pixel με όλες τις τιμές των χρωμάτων ίδιες παίρνοντας ως τιμή τον μέσο όρο των χρωμάτων (δηλαδή, $(120+70+96)/3 = 95$) και δίνοντάς αυτήν την τιμή και στα τρία χρώματα που απαρτίζουν το pixel, δηλαδή (95,95,95).

Για να αποφασίσετε την χρωματική απόσταση του χρώματος ενός pixel από το χρώμα που σας έδωσε ο χρήστης (και κατά συνέπεια να μην το κάνετε απόχρωση του γκρι αλλά να το διατηρήσετε) θα πρέπει να υλοποιήσετε έναν τρόπο υπολογισμού της απόστασης αυτής. Τον τρόπο που υλοποιήσατε θα πρέπει να τον περιγράψετε στην αναφορά σας με 2 διαφορετικές εικόνες ώστε να αποτιμήσετε τη συμπεριφορά του.

Παρακάτω φαίνεται ο πίνακας με τα ονόματα των πιο συνηθισμένων χρωμάτων και τις αντίστοιχες RGB τιμές τους.

Color	Όνομα	Χρώμα (R,G,B)
	Red	(255,0,0)
	Lime	(0,255,0)
	Blue	(0,0,255)
	Yellow	(255,255,0)
	Cyan / Aqua	(0,255,255)
	Magenta / Fuchsia	(255,0,255)

	Maroon	(128,0,0)
	Olive	(128,128,0)
	Green	(0,128,0)
	Purple	(128,0,128)
	Teal	(0,128,128)
	Navy	(0,0,128)

Η λειτουργία της μετατροπής σε καλλιτεχνικό ασπρόμαυρο καλείται με την παράμετρο `-bw` (από το `black & white`), το όνομα του χρώματος που θα διατηρηθεί, και το ποσοστό που αναφέρεται στην αρχή της ενότητας, χωρισμένα με κάτω παύλα. Για παράδειγμα η κλήση:

```
$> ./bip -bw_red_10 balloons balloons_out
```

θα πρέπει να δημιουργεί μία εικόνα με όνομα `balloons_out.bmp` η οποία θα είναι σε αποχρώσεις του γκρι εκτός από το 10% των `pixel` που είναι εγγύτερα στο κόκκινο. Παρακάτω δίνονται δύο αρχικές εικόνες και δύο παραδείγματα του καλλιτεχνικού ασπρόμαυρου στα οποία έχει διατηρηθεί το 10% των `pixel` που είναι εγγύτερα στο κόκκινο. Παρατηρείστε ότι λόγω του ποσοστού, δεν έχουν διατηρηθεί όλα τα κόκκινα `pixels` των εικόνων και ότι οι δύο εικόνες έχουν παραχθεί με δύο διαφορετικούς τρόπους υπολογισμού της απόστασης.



Αντίστοιχα η κλήση:

```
$> ./bip -bw_magenta_8 train train_magenta
```

θα πρέπει να δημιουργεί μία εικόνα με όνομα `train_magenta.bmp` η οποία θα είναι σε αποχρώσεις του γκρι εκτός από το 8% των `pixel` που είναι εγγύτερα στο φούξια.

2.5. Μείωση παλέτας χρωμάτων [15 μονάδες]

Ουσιαστικά κάθε βασικό χρώμα του RGB έχει 255 δυνατές τιμές με τις οποίες συμμετέχει και συνθέτει άλλα χρώματα. Ο χρήστης θα μπορεί να δώσει τον αριθμό τιμών που θέλει να έχει κάθε χρώμα (ίδιος για όλα τα χρώματα και πολλαπλάσιο του 2) και εσείς θα πρέπει να αντιστοιχίζετε τις τρέχουσες τιμές των χρωμάτων στη νέα παλέτα. Η λειτουργία της μείωσης της παλέτας χρωμάτων καλείται με την παράμετρο -p (από το palette) και τον αριθμό των χρωμάτων της παλέτας. Για παράδειγμα η κλήση:

```
$> ./bip -p4 balloons balloons_out
```

θα πρέπει να δημιουργεί μία εικόνα με όνομα balloons_out.bmp η οποία θα έχει μόνο 4 δυνατές τιμές ανά χρώμα και όλες οι τιμές στα διαστήματα [0,63], [64,127], [128,191], και [192,255] θα μετατραπούν στο κάτω όριο του αντίστοιχου διαστήματος, δηλαδή σε 0, 64, 128, και 192 αντίστοιχα, οπότε το pixel με χρώμα (120,50,200) θα μετατραπεί στο (64,0,192).

2.6. Αρθρωτή δομή αρχείων και χρήση makefiles [10 μονάδες]

Αποθηκεύστε τις συναρτήσεις επεξεργασίας εικόνας σε ξεχωριστό αρχείο χρησιμοποιώντας κατάλληλο/α αρχείο/α κεφαλής ή/και extern μεταβλητές. Δημιουργήστε ένα makefile το οποίο να είναι ευέλικτο (να χρησιμοποιεί μακροεντολές/μεταβλητές και αν γίνεται έμμεσους κανόνες εξάρτησης - αν δεν γίνεται εξηγήστε γιατί στην αναφορά) για να γίνεται αυτόματα η μεταγλώττιση. Τις μονάδες για το ερώτημα αυτό θα τις πάρετε μόνο αν έχετε υλοποιήσει τουλάχιστον ένα από τα ερωτήματα επεξεργασίας του αρχείου εικόνας.

2.7. Αναφορά [10 μονάδες]

Γράψτε μία σύντομη αναφορά 3-5 σελίδων (χωρίς το εξώφυλλο) στην οποία θα αναφέρετε ποια από τα ζητούμενα της άσκησης έχουν υλοποιηθεί και ποια όχι, να αναλύετε τη λύση σας όπου κρίνετε ότι χρειάζεται (π.χ., στους ελέγχους εγκυρότητας της εισόδου, στις συναρτήσεις επεξεργασίας της εικόνας, στο makefile, κ.λπ.). Στη αναφορά θα δίνετε επίσης λεπτομέρειες για την υλοποίησή σας, την εκτέλεσή της άσκησης, τις εισόδους που περιμένετε από το χρήστη σε κάθε περίπτωση, και για τις παραδοχές που κάνατε, όπως και μια μικρή επίδειξη χρήσης με screenshots από τα properties των αρχείων εικόνας αρχικά και μετά την επεξεργασία που κάνατε με το πρόγραμμά σας. Μη βάλετε τον κώδικά σας στην αναφορά!

3. Βαθμολόγηση

Για να διορθωθεί η εργασία και να πάρετε τις μονάδες που αντιστοιχούν σε κάθε ερώτημα, πρέπει να έχετε υλοποιήσει προφανώς την ανάγνωση/εγγραφή του αρχείου εικόνας ώστε να υπάρχει ορατό αποτέλεσμα της επεξεργασίας σας. Επίσης, φροντίστε να κάνετε τη διεπαφή χρήσης (από τη γραμμή εντολών) του προγράμματός σας φιλική προς το χρήστη και χρησιμοποιήστε επεξηγηματικά

μηνύματα και μηνύματα ολοκλήρωσης εργασιών ώστε να είναι κατανοητό τι πρέπει να κάνει ο χρήστης σε κάθε στιγμή ή τι περιμένει μετά την εντολή. Μια κακή διεπαφή χρήσης θα σας αφαιρέσει μονάδες!

4. Bonus

Δεν υπάρχει η δυνατότητα bonus για την άσκηση αυτή.

5. Σημαντικές σημειώσεις

- Για να πάρετε τις μονάδες που αντιστοιχούν σε κάθε ερώτημα, πρέπει να έχετε υλοποιήσει προφανώς και την ανάγνωση/εγγραφή του αρχείου εικόνας. Λύσεις που δεν υλοποιούν τις λειτουργίες: της ανάγνωσης/εγγραφής του αρχείου και της εξαγωγής πληροφοριών δεν θα γίνουν δεκτές και δεν θα βαθμολογηθούν! Επίσης απαραίτητη για τη βαθμολόγηση της άσκησής σας είναι η αναφορά.
- Η άσκηση θα τρέχει αποκλειστικά από ορίσματα γραμμής εντολών. Ασκήσεις που θα έχουν μενού και δεν θα λειτουργούν από ορίσματα γραμμής εντολών δε θα γίνονται δεκτές και δε θα βαθμολογούνται.
- Εξυπακούεται ότι θα χρησιμοποιήσετε δυναμική δέσμευση (και αποδέσμευση) μνήμης, μιας και δεν ξέρουμε εκ των προτέρων τι μέγεθος θα είναι το αρχείο εικόνας που θα πρέπει να επεξεργαστούμε.
- Μπορείτε να ορίσετε διαφόρων ειδών structs ομαδοποιώντας τα χρώματα ανά pixel, ή αφήνοντάς τα χωρίς ομαδοποίηση. Η πρώτη επιλογή έχει ευκολότερο χειρισμό στις πράξεις, ενώ η δεύτερη είναι ευκολότερη στο διάβασμα/γράψιμο από αρχείο.
- Δεν χρειάζονται όλες οι συναρτήσεις από τον κώδικα που σας δίνω. Χρειάζεστε μόνο τις συναρτήσεις readHeader(), readInfo(), και writeHeader(), writeInfo() για να διαβάσετε/γράψετε τα περιεχόμενα των δύο τμημάτων της εικόνας αντίστοιχα. Οι υπόλοιπες συναρτήσεις των αρχείων bmp.c και bmp.h χρησιμοποιούνται για την υλοποίηση των παραπάνω συναρτήσεων και δεν σας χρειάζονται.
- Τα αρχεία που θα χρειαστείτε θα τα βρείτε στο υποσύστημα “Εγγραφα” του eclass, μαζί με την παρούσα εκφώνηση της άσκησης. Στο ίδιο σημείο μπορείτε να βρείτε και διάφορες bmp εικόνες με 24 bit χρώματος, χωρίς συμπίεση για τις δοκιμές σας. Δώστε προσοχή στην εικόνα lena.bmp η οποία θεωρείται ως η πιο συχνά χρησιμοποιούμενη για δοκιμή αλγορίθμων επεξεργασίας εικόνας κι απεικονίζει ένα μέρος της φωτογραφίας της Σουηδής Lena Söderberg από την φωτογράφισή της για το περιοδικό Playboy τον Νοέμβριο του 1972. Μπορείτε φυσικά για τις δοκιμές σας να χρησιμοποιήσετε και όποια άλλη εικόνα θέλετε, αρκεί να είναι bitmap 24-bit και να μην χρησιμοποιεί συμπίεση/κωδικοποίηση (δείτε τις ιδιότητες της εικόνας για να σιγουρευτείτε).

Καλή δουλειά!