



ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

Τμήμα Πληροφορικής

ΕΠΛ 232 – Προγραμματιστικές Τεχνικές και Εργαλεία

Άσκηση 4: Ομαδική Συγγραφή Βιβλιοθήκης (.a) και Πελάτη για Στεγανογραφία σε Αρχεία Εικόνων

Διδάσκων: Ανδρέας Αριστείδου

Υπεύθυνος Άσκησης: Παύλος Αντωνίου & Πύρρος Μπράτσкас

Ημερομηνία Ανάθεσης: Τρίτη 10/11/2020

Ημερομηνία Παράδοσης: Παρασκευή, 27/11/2020 09:00 π.μ.

(να υποβληθεί σε ένα zip στο Moodle)

I. Στόχος Άσκησης

Στόχος αυτής της άσκησης είναι να σας επιτρέψει να βάλετε μαζί όλες τις γνώσεις και αρχές που αποκτήσατε μέσω του ΕΠΛ132. Το ειδικότερο θέμα της εργασίας αφορά τη **συγγραφή μιας βιβλιοθήκης επεξεργασίας εικόνων** τύπου BMP (Microsoft Bitmap) και η παρουσίαση των δυνατοτήτων της μέσω της υλοποίησης ενός **πελάτη**. Η υλοποίηση θα γίνει σε ομάδες των τριών (3) ατόμων, όπως αυτές έχουν ανακοινωθεί μέσω του Moodle. Για την υλοποίηση της άσκησης θα χρειαστεί να χρησιμοποιήσετε τα ακόλουθα στοιχεία:

Νέα Στοιχεία:

- Βιβλιοθήκη:** Καλείστε να υλοποιήσετε **κάθε μια από τις λειτουργίες της βιβλιοθήκης**, οι οποίες περιγράφονται στην ενότητα V αυτής της εκφώνησης, **ως ένα ξεχωριστό module (.c)** παρέχοντας **τα πρότυπα συναρτήσεων σε ένα ενιαίο αρχείο κεφαλίδας** (π.χ., δείτε σχετικό παράδειγμα string.h στη διάλεξη 16). Νοείται ότι η βιβλιοθήκη σας δύναται να έχει όλα άλλα modules (ζεύγη .c/.h) κρίνεται σκόπιμα. Εφαρμόστε το **κατάλληλο επίπεδο απόκρυψης πληροφοριών με χρήση PRIVATE ή PUBLIC**.
- Πελάτης:** Καλείστε να υλοποιήσετε **ένα πελάτη ο οποίος θα συνδέεται στατικά με τη βιβλιοθήκη για να παρουσιάζει τις λειτουργίες της**. Ο πελάτης θα ήταν καλό να αποτελείται από **ένα μονάχα .c αρχείο** (όλες οι υπόλοιπες λειτουργίες να ενσωματωθούν στην βιβλιοθήκη).
- Ανάπτυξη σε Ομάδες:** Για την **ανάπτυξη της ομαδικής αυτής εργασίας πρέπει να χρησιμοποιηθεί το σύστημα εκδόσεων git και το github**.
- Σχεδίαση Προγράμματος:** Προτρέπειται όπως η βιβλιοθήκη σας σχεδιαστεί από πάνω προς τα κάτω, αποφασίζοντας δηλαδή τα αρχεία κεφαλίδας συλλογικά και υλοποιώντας κάθε συνάρτηση ατομικά. Η **ορθότητα κάθε συνάρτησης της βιβλιοθήκης** (το οποίο ονομάζεται module στη περίπτωση μας), **θα πρέπει να ελεγχθεί και από τα δυο μέλη της ομάδας** αλλά θα πρέπει να υλοποιείται από **ένα μόνο μέλος της ομάδας** (του οποίου το **όνομα θα τοποθετηθεί και ως Author στο εν λόγω αρχείο**).
- Έλεγχος Προγράμματος:** Το **τελικό λογισμικό σας θα πρέπει να ελεγχθεί στατικά** (κατά τη μεταγλώττιση και με τον debugger) αλλά **και δυναμικά** (με valgrind για διαρροή μνήμης, profiler gprof για εύρεση συναρτήσεων που πρέπει να βελτιστοποιηθούν). **Νοείται ότι κάθε module του προγράμματος σας θα συνοδεύεται από τους σχετικούς οδηγούς χρήσης για να γίνει το σχετικό white-box testing**.
- GPL:** Για σκοπούς εξοικείωσης σας με την ορολογία του λογισμικού ανοικτού πηγαίου κώδικα, η **βιβλιοθήκη σας θα πρέπει να κάνει χρήση του GPL προοιμίου (preamble) σε κάθε αρχείο πηγαίου κώδικα της βιβλιοθήκης**. Ο πελάτης θα πρέπει να ανταποκρίνεται με το προοίμιο εάν δε δοθούν ορίσματα (βλέπε διάλεξη 18).
- Αξιολόγηση:** Η **αξιολόγηση της άσκησης θα στηριχτεί στα αναλυτικά κριτήρια που θα παρουσιαστούν στο τέλος αυτής της εκφώνησης**. Μεταξύ άλλων, βασικός στόχος της άσκησης είναι να **αξιολογηθεί η δομή και οργάνωση της βιβλιοθήκης σας αλλά και επίδοση του συνολικού προγράμματος σας** (π.χ., πόση ώρα θα χρειαστεί να διεκπεραιώσει μια ακολουθία από εντολές; Πόσο αποδοτικά χρησιμοποιείται η μνήμη, κτλ.)

Στοιχεία από Προηγούμενες Εργασίες:

1. Ανάπτυξη του προγράμματος σε **eclipse CDT IDE (προαιρετικό)**. Το πρόγραμμα πρέπει να μεταγλωττίζεται τόσο μέσω του eclipse όσο και μέσω της γραμμής εντολής με το **makefile**. Κάθε αντικείμενο (**module**) πρέπει να συμπεριλαμβάνει τους **σχετικό οδηγό χρήσης (driver functions)**, δείτε διάλεξη 12). Οι ομάδες μπορεί να χρησιμοποιήσουν τους οδηγούς χρήσης όπως επεξηγήθηκε στο μάθημα ή το CUnit.
2. **Σχόλια** και οδηγό σχολίων με χρήση του **doxygen** αλλά και διάγραμμα εξαρτήσεων αντικειμένων με χρήση του **graphviz**.

Οι λειτουργίες της βιβλιοθήκης σας και το αναμενόμενο αποτέλεσμα του πελάτη περιγράφονται αναλυτικότερα στην συνέχεια, αφού επεξηγηθεί πρώτα το πρόβλημα.

II. Εισαγωγή

Στεγανογραφία είναι η διαδικασία κατά την οποία αποκρύπτεται κάποια πληροφορία που πρέπει να αποσταλεί σε κάποιον παραλήπτη μέσα σε ένα μέσο. Ως μέσο μπορεί να οριστεί οποιοδήποτε υλικό αντικείμενο ή άυλο όπως είναι τα αρχεία ενός ηλεκτρονικού υπολογιστή. Ο σκοπός της στεγανογραφίας είναι η αποστολή της επιθυμητής πληροφορίας κρυμμένη μέσα στο μέσο, έτσι ώστε να μην γίνει αντιληπτή από ανεπιθύμητα άτομα, αλλά μόνο από τον παραλήπτη για τον οποίο προορίζεται.

Σε αντίθεση με την στεγανογραφία, υπάρχει και η σχετική αλλά διαφορετική έννοια της κρυπτογραφίας, η οποία ασχολείται με τη μελέτη της ασφαλούς επικοινωνίας. Στην κρυπτογραφία ένα μήνυμα κωδικοποιείται με τέτοιο τρόπο ώστε να γίνεται υπολογιστικά ακριβό για οποιοδήποτε τρίτο άτομο, πλην τον **καθορισμένο παραλήπτη ο οποίος διαθέτει το κλειδί αποκρυπτογράφησης**, να καταλάβει το περιεχόμενο του μηνύματος.

Στο παράδειγμα πιο κάτω βλέπουμε αριστερά ένα κείμενο και δεξιά τον τρόπο που μπορούμε να κρύψουμε ένα μυστικό μήνυμα μέσα σε αυτό. Κάθε λέξη του μυστικού μηνύματος εξάγεται παίρνοντας το πρώτο γράμμα κάθε λέξης της κάθε γραμμής του μηνύματος.

```
Big rumble in New Guinea.
The war on
Celebrity acts should end soon.
Over four
Big ecstatic elephants replicated.
```

```
Big rumber in New Guinea.
The war on
Celebrity acts should end soon.
Over four
Big ecstatic elephants replicated.
```

Το μυστικό κείμενο είναι: **Bring Two Cases Of Beer**

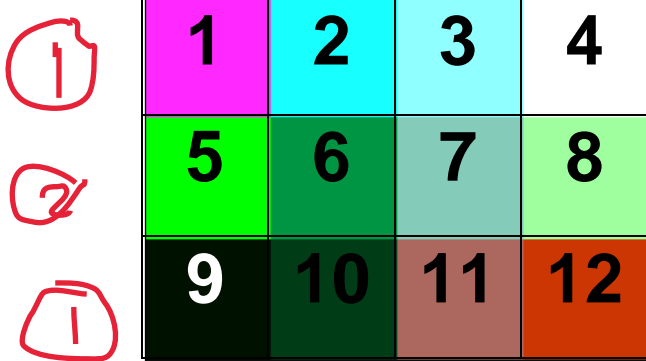
Τα **αρχεία υπολογιστών** (εικόνες, ήχοι, ακόμη και οι αποθηκευτικοί δίσκοι) **περιέχουν αχρησιμοποίητους ή ασήμαντους τομείς δεδομένων**. Η τέχνη της Στεγανογραφίας **εκμεταλλεύεται αυτές τις περιοχές, αντικαθιστώντας τις με χρήσιμες πληροφορίες**. Τα αρχεία αυτά μπορούν να ανταλλαχθούν χωρίς κανένας να ξέρει τι βρίσκεται πραγματικά μέσα τους. Έτσι, μια φωτογραφία μας μπορεί εύκολα να περιέχει μια **προσωπική επιστολή σε έναν φίλο/φίλη** και η πληροφορία αυτή να μην είναι ορατή στο γυμνό μάτι. Η Στεγανογραφία μπορεί επίσης να χρησιμοποιηθεί για να τοποθετήσει ένα κρυμμένο "εμπορικό σήμα", είναι αυτό που αποκαλείται **Watermarking**. Θα μπορούσε επίσης ένας εκτυπωτής να αλλάζει ελαφρώς κάποια pixels σε μια εκτυπωμένη σελίδα για να περιλάβει την ημερομηνία εκτύπωσης και τον αριθμό παραγωγής του εκτυπωτή (serial number). Για την εργασία αυτή, **η μέθοδος της στεγανογράφησης θα εφαρμοστεί στις εικόνες τύπου bitmap**. Στα επόμενα κεφάλαια της εκφώνησης, δίνεται η δομή των εικόνων bitmap, και ο τρόπος με τον οποίο γίνεται η Στεγανογραφία εικόνων.

III. Εικόνες Bitmap

Γνωρίζουμε ότι τα **αρχεία εικόνων** σε ένα υπολογιστή (π.χ., jpeg, tiff, gif, bmp, png, κτλ.) **αποθηκεύουν την πληροφορία σε δυαδική μορφή** (παρά σε αρχεία χαρακτήρων ASCII που είδατε μέχρι στιγμής). Τα

αρχεία με κατάληξη **.BMP** (ή **.DIB – Device Independent Bitmap**), είναι μια ειδική κατηγορία εικόνων που αναπτύχθηκε από την Microsoft. Η απλή εσωτερική δομή των αρχείων αυτών τα καθιστούν κατάλληλα για την διδασκαλία βασικών αρχών που έχετε διδαχθεί στα πλαίσια του μαθήματος μας.

Γνωρίζουμε ότι κάθε εικόνα αποτελείται από ένα αριθμό **εικονοστοιχείων (pixels)**. Θεωρήστε για παράδειγμα τον 3x4 πίνακα από pixels της εικόνας 1. Αν δώσουμε κάποιο χρώμα σε κάθε τετράγωνο τότε θα δημιουργηθεί κάποια εικόνα. Αν είχαμε ένα μεγαλύτερο πίνακα και **μια μεγάλη παλέτα χρωμάτων**, τότε θα μπορούσαμε να φτιάξουμε μια εικόνα όπως αυτή που παρουσιάζεται στην Εικόνα 2.



Εικόνα 1



Εικόνα 2

Βασικό χαρακτηριστικό μιας εικόνας είναι το **βάθος χρώματος (color depth)**. Με τον όρο αυτό αναφερόμαστε στο **πλήθος των διαφορετικών χρωμάτων από τα οποία μπορούν να επιλεγούν τα χρώματα ενός pixel**. Οι τυπικές περιπτώσεις είναι οι εξής:

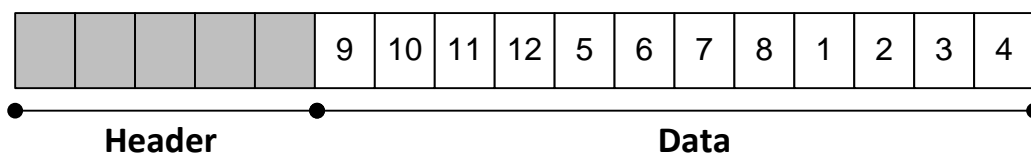
Βάθος χρώματος	Πλήθος χρωμάτων
1 bit	2 διαθέσιμα χρώματα (2^1)
4 bits	16 διαθέσιμα χρώματα (2^4)
8 bits ή 1 byte	256 διαθέσιμα χρώματα (2^8)
16 bits ή 2 bytes	65.536 χρώματα (2^{16})
24 bits ή 3 bytes	16.777.216 χρώματα (2^{24})
32 bits ή 4 bytes	4.294.967.296 χρώματα (2^{32})

Το μέγεθος ενός αρχείου λοιπόν εξαρτάται τόσο από το πλήθος των pixels όσο και από τα bytes τα οποία απαιτούνται για να περιγραφεί το χρώμα του κάθε pixel. Έτσι μια εικόνα 800x600 με βάθος χρώματος 24bit, έχει 800x600 pixels και κάθε pixel χρησιμοποιεί 3 bytes για να καθορίσει το χρώμα του, επομένως η εικόνα αυτή στη μνήμη (και στο δίσκο) θα χρησιμοποιεί τουλάχιστον $800 * 600 * 3 \text{ bytes} = 1.37\text{MB}$. Η λέξη “τουλάχιστον” δηλώνει ότι πέραν από τα χρήσιμα δεδομένα (data) της εικόνας, μαζί με την εικόνα αποθηκεύεται επίσης η επιπλέον πληροφορία της κεφαλίδας (header). Η κεφαλίδα αποθηκεύει διάφορες μέτα-πληροφορίες οι οποίες χαρακτηρίζουν τα pixels τα οποία ακολουθούν.

Για παράδειγμα το αρχείο της εικόνας 1 θα αποθηκευτεί στη δευτερεύουσα μνήμη στη μορφή που υποδεικνύεται στην Εικόνα 3 (το **DATA είναι ο Πίνακας της εικόνας 1, από κάτω-αριστερά προς πάνω-δεξιά, γραμμή-γραμμή**), ενώ τα σκιασμένα τετράγωνα υποδηλώνουν την κεφαλίδα της εικόνας.

Εικόνα 3

3x4.bmp



Όπως βλέπουμε, τα pixels (δηλ., το DATA) ακολουθούν τις πληροφορίες του HEADER. Όλοι οι τύποι **δυναμικών αρχείων** (.doc, .jpg,....) **έχουν κάποιο είδος header**. Συνεπώς με την ολοκλήρωση αυτής της

εργασίας θα έχετε εκτιμήσει πολύ καλά τι χρειάζεται για να επεξεργαστείτε πραγματικά αρχεία δεδομένων σε ένα πρόγραμμα.

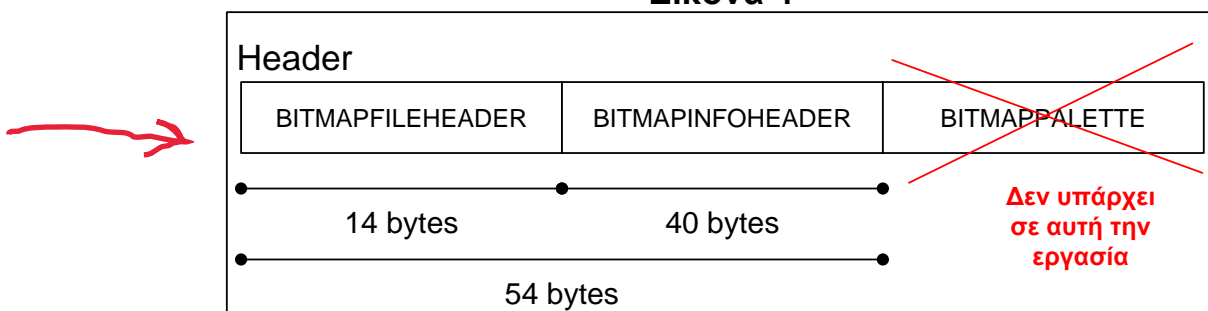
Α. Δομή του Header σε Αρχεία BMP

Ας δούμε τώρα αναλυτικότερα το Header μέρος ενός BMP αρχείου. Όπως φαίνεται στο σχήμα της Εικόνας 4, το Header διαιρείται σε 3 μέρη, κάθε ένα εκ' των οποίων περιέχει κάποια συνοδευτική πληροφορία για τη συγκεκριμένη εικόνα (μέτα-πληροφορία). Τα τρία μέρη **BITMAPFILEHEADER**, **BITMAPINFOHEADER** και **BITMAPPALETTE** παρουσιάζονται αναλυτικότερα στη συνέχεια. Το **BITMAPFILEHEADER** περιέχει πληροφορίες για το αρχείο (π.χ., το μέγεθος του αρχείου σε bytes), το **BITMAPINFOHEADER** πληροφορίες για την εικόνα (π.χ., χρώμα, μέγεθος, κτλ) και το **BITMAPPALETTE** περιέχει ένα πίνακα χρωμάτων.

Σε αυτή την εργασία θα μας απασχολήσουν μόνο τα πρώτα δυο κομμάτια του Header ενώ το **BITMAPPALETTE** θα το αγνοήσουμε. Αυτό διότι το **BITMAPPALETTE** υπάρχει μόνο στις εικόνες με βάθος χρώματος μέχρι 8-bits. Οι εικόνες με βάθος χρώματος 16 bits και μεγαλύτερο δεν έχουν **BITMAPPALETTE**. Στις εικόνες λοιπόν με λίγα χρώματα, το **BITMAPPALETTE** περιγράφει ποια χρώματα χρησιμοποιεί η εικόνα (π.χ., σε μια εικόνα με 256 χρώματα, θα πρέπει να δηλώσουμε ποια είναι αυτά τα 256 χρώματα από το σύνολο των χρωμάτων που μπορεί να απεικονίσει ο υπολογιστής, π.χ., το byte έχει τα ακόλουθα bits R,R,R,G,G,G,B,B, όπου R=Red, G=Green και B=Blue). Στην παρούσα εργασία θα ασχοληθούμε μόνο με εικόνες βάθους χρώματος 24 bits στις οποίες η κατανομή χρωμάτων είναι πάντα η ίδια για το R, G, B, δηλαδή 3 bytes συνολικά.

Τα **BITMAPFILEHEADER** και **BITMAPINFOHEADER** έχουν συνολικό μέγεθος 54 bytes, όπως φαίνεται και στο παρακάτω σχήμα.

Εικόνα 4



Οι επόμενοι πίνακες περιγράφουν αναλυτικά τα πεδία των headers. Εσείς θα κληθείτε να επεξεργαστείτε τα πεδία αυτά για να υλοποιήσετε τα ζητούμενα της άσκησης αυτής. Ειδικότερα θα πρέπει να ορίσετε τις κατάλληλες δομές (struct) λαμβάνοντας υπόψη θέματα ευθυγράμμισης μνήμης. Για ευκολία αναφοράς καλέστε να χρησιμοποιήσετε τις ακόλουθες δηλώσεις:

```
typedef unsigned char byte;
typedef unsigned short int word;
typedef unsigned int dword;
```

Τα σκιασμένα πεδία δηλώνουν τα πεδία τα οποία μάλλον δεν θα χρειαστείτε να επεξεργαστείτε (εάν και πρέπει να είναι μέρος των δομών σας).

BITMAPFILEHEADER:

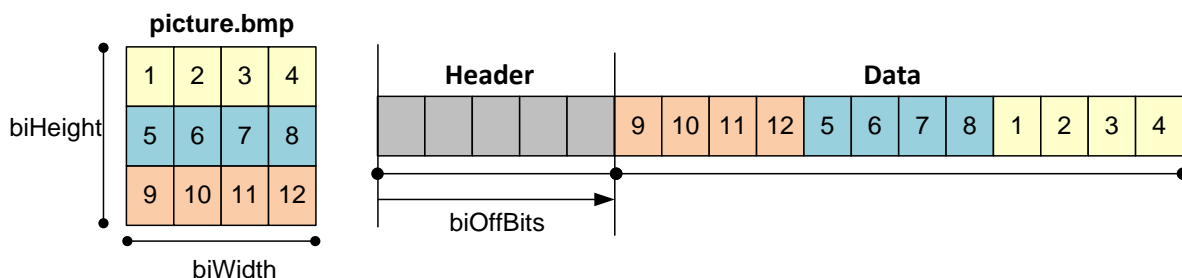
Bytes	Όνομα	Χρήση
1	bfType1	Must always be set to 'BM' to declare that this is a .bmp file (i.e., bfType1='B' and bfType2='M'. Based on these two fields you can identify if this is Bitmap file or not.
1	bfType2	
4	bfSize	Specifies the size of the file (including padding) in bytes
2	bReserved1	Usually set to zero.
2	bReserved2	Usually set to zero.
4	bfOffBits	Specifies the offset from the beginning of the file to the bitmap data.

BITMAPINFOHEADER:

Bytes	Όνομα	Χρήση
4	biSize	Specifies the size of the BITMAPINFOHEADER structure, in bytes.
4	biWidth	Specifies the width of the image, in pixels.
4	biHeight	Specifies the height of the image, in pixels.
2	biPlanes	Specifies the number of planes of the target device, usually set to zero.
2	biBitCount	Specifies the number of bits per pixel.
4	biCompression	Specifies the type of compression, usually set to zero (no compression). Need to provide an error if image is compressed.
4	biSizeImage	Specifies the size of the image data, in bytes. If there is no compression, it is valid to set this member to zero.
4	biXPelsPerMeter	Specifies the horizontal pixels per meter on the designated target device, usually set to zero.
4	biYPelsPerMeter	Specifies the vertical pixels per meter on the designated target device, usually set to zero.
4	biClrUsed	Specifies the number of colors used in the bitmap, if set to zero the number of colors is calculated using the biBitCount member.
4	biClrImportant	Specifies the number of color that are 'important' for the bitmap, if set to zero, all colors are important.

B. Δομή του DATA σε Αρχεία BMP

Τώρα θα δούμε πώς αποθηκεύονται τα DATA (δηλαδή τα πραγματικά pixels μιας εικόνας) μέσα στο αρχείο (σε μη-συμπίεσμένη μορφή, ενώ η συμπίεσμένη μορφή δεδομένων θα αγνοηθεί για την εργασία). Η περιοχή DATA, αποθηκεύει σειριακά τις γραμμές της εικόνας, από το **κάτω-αριστερά άκρο της εικόνας προς το άνω-δεξιά άκρο**. Σχηματικά, η αποθήκευση φαίνεται στο πιο κάτω σχήμα:

Εικόνα 5

Σημειώστε ότι το **biWidth** (μέσα στο BITMAPINFOHEADER) **δίδει το πλάτος μιας εικόνας ενώ το biHeight** (μέσα στο BITMAPINFOHEADER) **το ύψος της εικόνας**. Επίσης το **biBitCount** (μέσα στο BITMAPINFOHEADER) **δίδει το βάθος χρώματος** (σε αυτή την άσκηση όλες οι εικόνες είναι 24 bits – δηλαδή κάθε pixel θα καταλαμβάνει 3 bytes). Σημειώστε επίσης ότι το πεδίο **biOffBits** (μέσα στο BITMAPFILEHEADER) σας **δίνει την ακριβή θέση** (σε σχέση με την αρχή του αρχείου) **απ' όπου ξεκινούν τα Data, εάν και δεν χρειάζεται μάλλον να χρησιμοποιηθεί**.

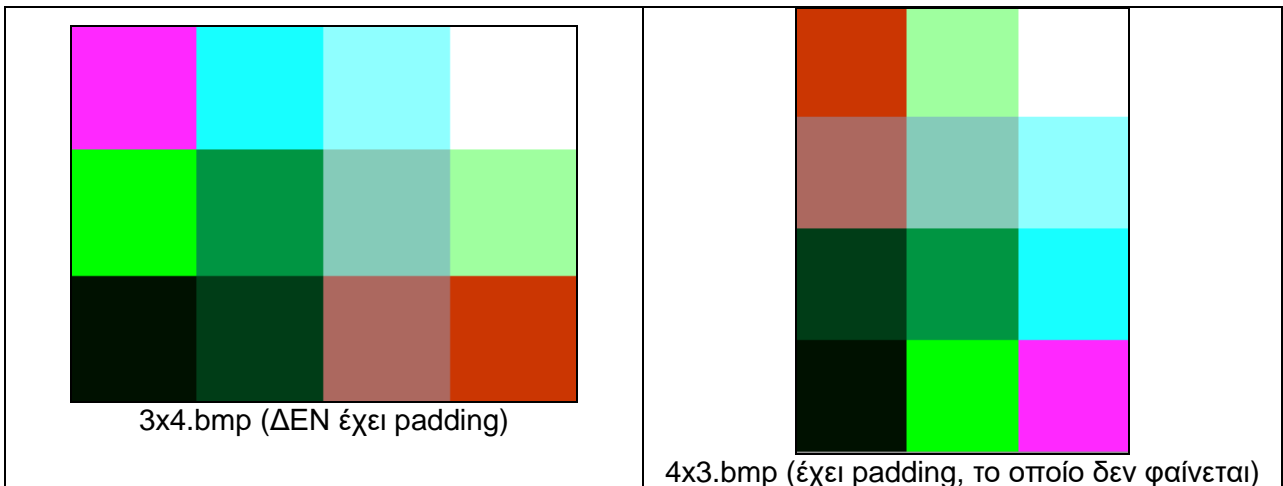
Περιπλοκές

Όπως αναφέραμε, **μια εικόνα έχει biWidth αριθμό pixels σε κάθε γραμμή**. Εάν το **$(biWidth * 3) \% 4$** αφήνει υπόλοιπο μηδέν, τότε κάθε γραμμή της εικόνας δεν έχει τίποτα άλλο στο τέλος της. Εναλλακτικά, εάν **$(biWidth * 3) \% 4$** αφήνει υπόλοιπο διάφορο του μηδέν, τότε **περιέχει επιπλέον $4 - ((3 * biWidth) \% 4)$ κενά pixels στο τέλος κάθε γραμμής, τα οποία ονομάζονται padding**. Κάθε padding pixel αποτελείται από 3 κενά bytes (δηλ., έχει τη τιμή NUL – 0x00). Σημειώστε ότι τα **padding pixels δεν είναι μέρος του**

περιεχομένου της εικόνας αλλά παρουσιάζονται στο αρχείο της εικόνας για λόγους ευθυγράμμισης που δεν θα μας απασχολήσουν.

Για να κατανοήσετε την πιο πάνω περιπλοκή θεωρήστε τις ακόλουθες δυο περιπτώσεις:

- Η πιο κάτω εικόνα **3x4.bmp (αριστερά)** έχει 4 pixels σε κάθε γραμμή. Εφόσον το $(biWidth*3)\%4$ αφήνει υπόλοιπο μηδέν, αυτό σημαίνει ότι το αρχείο **ΔΕΝ** θα έχει οποιοδήποτε **padding pixel** σε κάθε γραμμή.
- Η πιο κάτω εικόνα **4x3.bmp (δεξιά)** από την άλλη έχει 3 pixels σε κάθε γραμμή. Εφόσον το $(biWidth*3)\%4$ αφήνει υπόλοιπο ίσο με 1, αυτό σημαίνει ότι η κάθε γραμμή θα έχει 4- $((biWidth*3)\%4)$, δηλαδή **1 padding pixel** (δηλαδή 3 επιπλέον bytes).



Κατανόηση Δομής Δυαδικών Αρχείων με Hexdump


Η πιο κάτω εκτέλεση της εντολής `hexdump` δείχνει το περιεχόμενο του **4x3.bmp** αρχείου σε bytes: η **πρώτη στήλη** δείχνει τη διεύθυνση της γραμμής σε δεκαεξαδική αναπαράσταση (π.χ., 00000020 σε δεκαεξαδική αναπαράσταση είναι το 32 σε δεκαδική αναπαράσταση), η **δεύτερη στήλη** δείχνει ένα-ένα τα bytes του αρχείου (για χάρη παρουσίασης κάθε γραμμή περιέχει 16 bytes) και η **τρίτη στήλη** δείχνει κάθε ένα από τα 16 bytes της γραμμής ως να ήταν ASCII χαρακτήρες (στη περίπτωση μας μόνο οι πρώτοι 2 χαρακτήρες, BM είναι πραγματικά ASCII χαρακτήρες, τα υπόλοιπα είναι ακολουθίες δυάδων byte, word, dword).

`$hexdump -C 4x3.bmp`

ΣΤΗΛΗ 1	ΣΤΗΛΗ 2	ΣΤΗΛΗ 3
00000000	42 4d 68 00 00 00 00 00 00 00 36 00 00 00 28 00	BMh.....6...(.
00000010	00 00 03 00 00 00 04 00 00 00 01 00 18 00 00 00
00000020	00 00 32 00 00 00 12 0b 00 00 12 0b 00 00 00 00	..2.....
00000030	00 00 00 00 00 00 00 11 00 00 ff 00 ff 28 ff 00(
00000040	00 00 17 3d 00 42 95 00 ff ff 17 00 00 00 5f 68	...=.B....._h
00000050	ac b9 cb 85 ff ff 8f 00 00 00 02 36 cb 9f ff 9f6....
00000060	ff ff ff 00 00 00 00 00
00000068		

Από την πιο πάνω έξοδο της `hexdump` παρατηρούμε τα ακόλουθα: **(α)** τα πρώτα 54 bytes, τα οποία είναι σκιασμένα με γκριζό, για χάρη παρουσίασης, αναφέρονται στο header της εικόνας. **(β)** Το header ακολουθείται από 4 γραμμές pixels, η κάθε μια εκ των οποίων γραμμές έχει 3 pixel (υπογραμμισμένα) και 1 pixel padding (3 bytes σκιασμένο και κόκκινο). Συνολικά δηλαδή, σε κάθε γραμμή υπάρχουν 16 pixels, εκ των οποίων τα χρήσιμα είναι μόνο τα 12 υπογραμμισμένα. **(γ)** Το πρώτο pixel (από κάτω αριστερά προς κάτω δεξιά) της εικόνας 4x3 είναι αυτό που μοιάζει με "μαύρο" (001100), το δεύτερο αυτό που μοιάζει με "πράσινο ανοικτό" (00ff00) και το τρίτο αυτό που μοιάζει με "ροζ" (ff28ff). Παρατηρήστε ότι κάθε pixel αποτελείται από 24 bits (8bits κόκκινο, 8 bits πράσινο και 8 bits μπλέ), δηλαδή προκύπτει η πιο κάτω ακολουθία 24 bits: **RRRRRRRRGGGGGGGGBBBBBBBB**.

Συμβουλή: Χρησιμοποιήστε το εργαλείο GIMP (είναι δωρεάν εάν το θέλετε για τον Η/Υ σας!), για να δείτε τις εικόνες που βρίσκονται μαζί με την εκφώνηση (για τις εικόνες 3x4 και 4x3 κάνετε zoom έτσι ώστε να

μπορείτε να επιλέξετε ένα pixel και να δείτε τι ακριβώς χρώμα έχει, κάνοντας χρήση του color picker , π.χ., δείτε <http://docs.gimp.org/en/gimp-tool-color-picker.html>)

Το αρχείο εισόδου που βρίσκονται στην ιστοσελίδα περιέχουν τόσο εικόνες με padding όσο και εικόνες χωρίς padding. Βεβαιωθείτε ότι το πρόγραμμα σας εκτελείται ορθά για όλες τις εικόνες που δίνονται. Τέλος σημειώστε ότι τα padding δεν είναι ορατά με το συμβατικό gimp viewer αλλά μόνο με το hexdump.

IV. Στεγανογραφία

Για να κρύψουμε εικόνα ή κείμενο μέσα σε μια άλλη εικόνα bmp, θα χρησιμοποιήσουμε το γεγονός ότι το ανθρώπινο μάτι δυσκολεύεται να διακρίνει τις αποχρώσεις των χρωμάτων που είναι πολύ κοντά το ένα στον άλλο. Αν λάβουμε υπόψη δύο γειτονικά pixel σε μια εικόνα, των οποίων οι αναπαραστάσεις σε δυαδική μορφή είναι οι εξής:

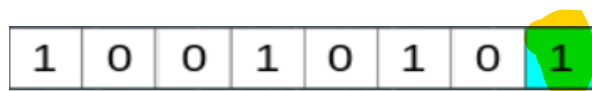
```
Pixel1 = (10010110(2), 11010101(2), 00101100(2))
Pixel2 = (10010110(2), 11010100(2), 00101100(2))
```

Είναι λοιπόν σχεδόν αδύνατο να ανιχνευθεί μια διαφορά χρώματος με γυμνό μάτι μεταξύ των δύο χρωμάτων. Ωστόσο, διαφέρουν το ένα από το άλλο, αφού τα bytes του πράσινου χρώματος (δεύτερο byte) είναι διαφορετικά. Ως εκ τούτου, μπορούμε να συμπεράνουμε ότι η αλλαγή των λιγότερο σημαντικών bits στη συνιστώσα ενός pixel, δεν έχει σχεδόν καμία επίπτωση στην οπτική αντίληψη μας για την εικόνα.

Για την εργασία αυτή, θα πρέπει να επεξεργαστείτε μέσω στεγανογραφίας εικόνες τύπου bitmap. Συγκεκριμένα, πρέπει να υλοποιήσετε ένα στεγανογραφικό σύστημα υλοποιώντας 8 λειτουργίες μετασχηματισμού δεδομένων. Οι δυο πρώτες λειτουργίες (Encrypting/Decrypting) θα παρέχουν τη δυνατότητα να κρύψουν/αποκρύψουν μια μυστική εικόνα εντός των pixels μιας άλλης εικόνας την οποία θα την αποκαλούμε εικόνα κάλυμμα. Οι άλλες δυο λειτουργίες, θα παρέχουν τη δυνατότητα να κρύψουν/αποκρύψουν ένα κείμενο μέσα σε μια εικόνα. Οι τελευταίες δυο λειτουργίες αφορούν την δημιουργία εικόνας από κείμενο και μετά από αυτό μπορείτε να χρησιμοποιήσετε τις δυο πρώτες λειτουργίες για να κρύψετε την εικόνα κείμενο.

A. Κρυπτογράφηση/Αποκρυπτογράφηση εικόνας μέσα σε μια άλλη εικόνα

Αυτή η λειτουργία κρύβει μια μυστική εικόνα μέσα σε μια εικόνα κάλυμμα. Το αποτέλεσμα είναι μια κρυπτογραφημένη εικόνα. Για να γίνει αυτό, θα χρησιμοποιήσουμε την τεχνική της αλλαγής των bits μικρότερου βάρους (lowest bits). Σε ένα byte το bit μικρότερου βάρους είναι αυτό που βρίσκεται στην πρώτη θέση από τα δεξιά (το μπλε bit στην εικόνα 6).



Εικόνα 6 Ένα byte

Όπως εξηγήσαμε στο Κεφάλαιο III, ένα pixel εικόνας bmp χρησιμοποιεί 3 bytes, ένα για το κόκκινο χρώμα, ένα για το πράσινο και ένα για το μπλε χρώμα. Το καθένα από αυτά, έχει μια τιμή μεταξύ 0...255 από την οποία εξαρτάται η ένταση του ανάλογου χρώματος (Red, Green, Blue).

Η τεχνική της αλλαγής των μικρότερου βάρους bits, είναι η ακόλουθη:

- Η εικόνα κάλυμμα και η μυστική εικόνα πρέπει να έχουν τις ίδιες διαστάσεις.
- Για κάθε byte του κάθε χρώματος, αν αλλάξουμε τα δυο δεξιότερα bits αυτό δεν θα επηρεάσει και πολύ την ένταση του χρώματος.
- Η ιδέα της τεχνικής είναι να αντικαταστήσουμε αυτά τα δυο bits της εικόνας κάλυμμα με τα bits μεγαλύτερου βάρους (τα δυο πρώτα από τα αριστερά) από την μυστική εικόνα.
- Η τεχνική αντί για δυο, μπορεί να αντικαταστήσει 1, 3 ή και 4 bits. Το πλήθος των αλλαγών, θα επηρεάσει την ποιότητα της μυστικής εικόνας όταν θελήσουμε να την επανακτήσουμε.

Παράδειγμα:

Έστω η πρωτότυπη εικόνα κάλυμμα « ο λαγός στο χιόνι » (αριστερά) και η πρωτότυπη μουσική εικόνα « το F15 » (δεξιά) :



Εάν αντικαταστήσουμε το **1 bit** μικρότερου βάρους της εικόνας κάλυμμα με το **1 bit** μεγαλύτερου βάρους της μουσικής εικόνας, και μετά επανακτήσουμε την μουσική εικόνα, το αποτέλεσμα θα είναι η εικόνα 7 (δεξιά) (αριστερά η εικόνα κάλυμμα μετά την εμφύτευση σε αυτή της μουσικής εικόνας):



Εικόνα 7 Αλλαγή ενός bit

Εάν αντικαταστήσουμε **2 bit** μικρότερου βάρους της εικόνας κάλυμμα με τα **2 bit** μεγαλύτερου βάρους της μουσικής εικόνας, και μετά επανακτήσουμε την μουσική εικόνα, το αποτέλεσμα θα είναι η εικόνα 8 (δεξιά) (αριστερά η εικόνα κάλυμμα μετά την εμφύτευση σε αυτή της μουσικής εικόνας):



Εικόνα 8 Αλλαγή δυο bits

Εάν αντικαταστήσουμε τα **4 bit** μικρότερου βάρους της εικόνας κάλυμμα με τα **4 bit** μεγαλύτερου βάρους της μουσικής εικόνας, και μετά επανακτήσουμε την μουσική εικόνα, το αποτέλεσμα θα είναι η εικόνα 9 (δεξιά) (αριστερά η εικόνα κάλυμμα μετά την εμφύτευση σε αυτή της μουσικής εικόνας):



Εικόνα 9 Αλλαγή τεσσάρων bits

Στα παραπάνω παραδείγματα μπορείτε να διακρίνετε ότι η εικόνα κάλυμμα μετά την εμφύτευση σε αυτή της μυστικής εικόνας δεν διαφέρει και πολύ από το πρωτότυπο της εικόνας κάλυμμα. Όμως, η ένταση των χρωμάτων της, αλλάζει προς το χειρότερο εάν αλλάξουμε περισσότερα bits της, με bits από την μυστική εικόνα. Το αντίθετο συμβαίνει με την μυστική εικόνα, η ποιότητα της οποίας είναι καλύτερη στην επανάκτηση εάν η αλλαγή αφορά περισσότερα bits. Αυτό συμβαίνει καθώς κατά την επανάκτηση μπορούμε να επανακτήσουμε περισσότερα bits που ανήκουν στην πρωτότυπη μυστική εικόνα. Επειδή αυτά είναι τα μεγαλύτερου βάρους bits η τιμή του χρώματος είναι μεγαλύτερη (προς το 255), τότε η ένταση των χρωμάτων της είναι καλύτερη. Κατά την επανάκτηση της μυστικής εικόνας, επανακτούμε μόνο τα bits της πρωτότυπης μυστικής εικόνας που εμφυτεύσαμε στην πρωτότυπη εικόνα κάλυμμα. Τα άλλα bits είναι 0, γι' αυτό και η αλλαγή στην ένταση των χρωμάτων.

B. Κρυπτογράφηση/Αποκρυπτογράφηση κειμένου μέσα σε μια εικόνα

Για αυτήν την ενότητα της εργασίας, θα υλοποιήσετε δυο λειτουργίες οι οποίες κρύβουν και εξάγουν κείμενο μέσα/από σε μια εικόνα bmp. Η διαδικασία κωδικοποίησης είναι η εξής:

Έστω ένα μήνυμα m το οποίο θέλουμε να κρύψουμε σε μια εικόνα και m_i το i -οστό byte αυτού του μηνύματος, και $[m_i]_j$ το j -οστό bit αυτού του i -οστού byte (τα bytes είναι αριθμημένα ξεκινώντας από το 0, και τα bits αριθμούνται από το 7 για το πιο σημαντικό bit, στο 0 για το λιγότερο σημαντικό bit). Είναι εύκολο τότε να ορίσουμε μια σειρά u_n η οποία περιέχει τα ακόλουθα bits του μηνύματος m :

$$u_n = \begin{cases} \left\lfloor \frac{m_{\frac{n}{8}}}{8} \right\rfloor_{(7-(n \% 8))} & \text{εάν } n \in [0 \dots 8 \times \text{strlen}(m)] \\ 0 & \text{εάν } n \notin [0 \dots 8 \times \text{strlen}(m)] \end{cases}$$

Προσθέστε μια συνάρτηση `int getBit(char *m, int n);` στο πρόγραμμα η οποία υπολογίζει το u_n .

Για την κωδικοποίηση πρέπει να κρύψουμε την σειρά των bits u_n μέσα στα bytes των χρωμάτων της εικόνας.

Πίνακας μετάθεσης

Για να μην τοποθετήσουμε το μήνυμα m σε συνεχόμενα bytes της εικόνας bmp, θα καθορίσουμε μια συνάρτηση ικανή για να κατασκευάσει μεταθέσεις (permutation) βασιζόμενα σε μια παράμετρο που θα ονομάζεται κλειδί του συστήματος (system-key-integer). Η μετάθεση θα μας επιτρέψει να γνωρίζουμε ποια είναι τα bytes της εικόνας που θα αλλάξουμε, και με ποια σειρά.

Αυτή η συνάρτηση f χρησιμοποιεί ένα πίνακα N ακέραιων το i -οστό στοιχείο του οποίου έχει την τιμή $f(i)$.

Ορισμός συνάρτησης μετάθεσης:

Έστω $K = [0 \dots N-1]$ σύνολο ακέραιων τότε:

$$f : K \longrightarrow K$$

$$\forall (x, y) \in K^2, f(x) = f(y) \iff x = y$$

Για να επιτύχουμε το στόχο μας, ορίζουμε την συνάρτηση f ως την πιο απλή μετάθεση που είναι, δηλαδή:

$$f(x) = x.$$

Δηλαδή, στην i -οστή θέση του πίνακα θα δώσουμε την τιμή i . Μετά θα αλλάξουμε τυχαία τα στοιχεία του πίνακα χρησιμοποιώντας τον παρακάτω αλγόριθμο:



Κάνε N φορές
 Διάλεξε τυχαία δυο ακέραιους i και j
 Μετάτρεψε τους στο διάστημα $[0..N - 1]$
 Αλλάξε τα στοιχεία στις θέσεις i και j του πίνακα

Το πρόβλημα είναι ότι πρέπει να δημιουργήσουμε ξανά τον ίδιο πίνακα της συνάρτησης μετάθεσης ανά πάσα στιγμή εάν το χρειαστούμε. Για να κάνετε αυτό αρκεί να χρησιμοποιήσετε την συνάρτηση `srand(<system-key-integer>)` και μετά να καλέσετε την συνάρτηση `rand()` η οποία επιστρέφει ψευδο-τυχαίους αριθμούς. Το πρότυπο της συνάρτησης θα είναι:

```
int * createPermutationFunction(int N, unsigned int systemkey);
```

Εισαγωγή μηνύματος στην εικόνα

Η εισαγωγή των bits του μηνύματος στην εικόνα bmp μπορεί να γίνει με την ακόλουθη διαδικασία:

Για κάθε i από 0 στο $(1 + \text{strlen}(m)) \times 8$

- υπολόγισε $b = \text{getBit}(m, i)$
- υπολόγισε $o = \text{permutation}[i]$
- διέγραψε το bit μικρότερου βάρους του o -οστού byte του πίνακα των pixels
- αντικατέστησε αυτό το bit που διαγράφηκε με την τιμή b

Η τοποθέτηση $(1 + \text{strlen}(m)) \times 8$ bits στην εικόνα, επιτρέπει την απόκρυψη όλων των bits του μηνύματος m (δηλαδή $\text{strlen}(m) \times 8$ bits) συν το μηδέν (NUL character) για το τέλος της συμβολοσειράς.

Εάν ακολουθηθεί η αντιστροφή διαδικασία που περιγράφηκε πιο πάνω, με προηγούμενη γνώση το `<system-key-integer>`, τότε θα οδηγηθούμε από μια κρυπτογραφημένη εικόνα στο κείμενο m .

Γ. Δημιουργία εικόνας από κείμενο

Μια άλλη μέθοδος για την εισαγωγή κειμένου σε μια εικόνα είναι η χρησιμοποίηση της διαδικασίας που περιγράφεται στην ενότητα Α αυτού του κεφαλαίου, δηλαδή να κρύψουμε μια εικόνα μέσα σε μια άλλη εικόνα κάλυμμα.

Δημιουργία εικόνας από κείμενο

Για να γίνει αυτό, πρέπει πρώτα να δημιουργήσουμε μια εικόνα από το κείμενο που θέλουμε να κρύψουμε. Αυτό μπορεί να γίνει δημιουργώντας μια ασπρόμαυρη εικόνα χρησιμοποιώντας τα bits του κειμένου. Η διαδικασία είναι η ακόλουθη:

- Κάθε pixel της παραγόμενης εικόνας θα αντιστοιχεί σε ένα bit από την ASCII αναπαράσταση του κειμένου.

- Εάν το bit είναι 0, τότε το pixel θα είναι μαύρο εάν το bit είναι 1, τότε το pixel θα είναι φωτεινό (bright).
- Το πρώτο bit του πρώτου χαρακτήρα του κειμένου, θα καθορίσει το χρώμα του pixel στην πάνω αριστερή γωνία της εικόνας. Το δεύτερο bit του ίδιου χαρακτήρα θα καθορίσει το χρώμα του επόμενου pixel στην πρώτη στήλη. Τα εναπομείναντα bits του πρώτου χαρακτήρα και των επόμενων χαρακτήρων θα τοποθετηθούν στα pixels προς τα κάτω στην αριστερή στήλη της εικόνας μέχρι το τέλος της στήλης.
- Μετά, συνεχίζει η διαδικασία ξεκινώντας από το pixel στην κορυφή της δεύτερης στήλης και ούτω καθεξής.

Για την διαδικασία αυτή, εκτός από τα bits της συμβολοσειράς του κειμένου, χρειάζεται να λάβετε υπόψη και τις διαστάσεις της εικόνας. Η παραγόμενη εικόνα, θα είναι μια γκριζα εικόνα τέτοια ώστε η φωτεινότητα του pixel στη θέση (i, j) να ισούται με $128 \times \text{getBit}(m, \text{height} \times i + j)$ όπου height είναι το ύψος της εικόνας.

Παράδειγμα:

Έστω ο πίνακας των bits `[0 1 0 0 0 0 1 0 1 0 0 0 1 1]` από τον οποίο θέλουμε να παράγουμε μια εικόνα 4x4. Τότε, τα pixels της εικόνας θα έχουν τιμές:

0	0	0	0
128	0	128	0
0	0	0	128
0	128	0	128

Εάν τελειώσουν τα bits και παραμένουν pixels προς συμπλήρωση, τότε θέτουμε τις τιμές των εναπομεινάντων pixels σε 0. Για παράδειγμα, εάν από τον πίνακα των bits array `[0 1 0 0 0 0 1 0 1 0 0 0 0 1 1]` θέλουμε να παράγουμε μια εικόνα 5x4, τότε τα pixels της εικόνας θα έχουν τις τιμές:

0	0	0	128
128	0	0	0
0	128	0	0
0	0	0	0
0	128	128	0

Εάν δεν υπάρχουν πια pixels και υπάρχουν ακόμα διαθέσιμα bits του κειμένου, τότε σταματάμε την κωδικοποίηση του πίνακα των bits. Αυτό σημαίνει ότι η αποκωδικοποίηση θα μας δώσει μόνο ένα μέρος του αρχικού κειμένου.

Για να έχετε μια ολοκληρωμένη εικόνα bmp, πρέπει να δώσετε τιμές και στα άλλα πεδία της εικόνας (header κλπ).

Για την αποκρυπτογράφηση/επανάκτηση του κειμένου πρέπει να ακολουθηθούν τα ίδια βήματα σε αντίστροφη σειρά. Πρώτα θα δημιουργηθεί μια σειρά από bits, μετά πρέπει να γίνει η αποκωδικοποίηση παίρνοντας τα bits ανά οκτάδες για να υπολογιστούν οι χαρακτήρες ASCII του κειμένου.

V. Ζητούμενα Άσκησης

Έστω ότι `bmplib.a` είναι η βιβλιοθήκη που θα δημιουργήσετε και `bmpSteganography.c` ο πελάτης, τότε `bmpSteganography` είναι το εκτελέσιμο αρχείο που παράγεται συνδέοντας το `bmplib.a` με το `bmpSteganography.c` κατά την μεταγλώττιση (υποδειγματικές εικόνες έχουν αναρτηθεί στο `as5-supplementary.zip`). Πιο κάτω περιγράφεται η αναμενόμενη λειτουργία της βιβλιοθήκης κάνοντας αναφορά στις λειτουργίες της μέσω υποδειγματικών εκτελέσεων του πελάτη.

A. Πελάτης

Η γενική μορφή εκτέλεσης του πελάτη είναι η ακόλουθη:

```
$./bmpSteganography <-option> image1.bmp [image2.bmp image3.bmp ...]
```

όπου **option** σχετίζεται με μια επί μέρους λειτουργία της βιβλιοθήκης (π.χ., -list εκτυπώνει το header, που θα δούμε σε λίγο) και στη συνέχεια ακολουθεί ένας απροσδιόριστος αριθμός από ονόματα αρχείων.

Ενδεχόμενα Λάθη:

- **Δεν ορίζεται option ή δεν δίνεται το όνομα τουλάχιστο μιας εικόνας:** Πρέπει να εκτυπώνεται το GPL προοίμιο και στη συνέχεια να δίνεται το σχετικό μήνυμα λάθους.
- **Μια εικόνα δεν είναι .bmp ή δεν είναι 24-bit ή είναι συμπιεσμένη:** Η εικόνα πρέπει να αγνοείται (η αναγνώριση να γίνει από το bfType και όχι από το extension του αρχείου).

B. Βιβλιοθήκη

Λειτουργία 1: Εξαγωγή Μέτα-Πληροφοριών

Το όρισμα **-list** αφορά τη εκτύπωση των στοιχείων του header μιας προσδιορισμένης εικόνας σε μορφή που ακολουθεί.

```
$./bmpSteganography -list 4x3.bmp image2.bmp
```

```
BITMAP_FILE_HEADER
=====
bfType: BM
bfSize: 104
bfReserved1: 0
bfReserved2: 0
bfOffBits: 54
```

```
BITMAP_INFO_HEADER
=====
biSize: 40
biWidth: 3
biHeight: 4
biPlanes: 1
biBitCount: 24
biCompression: 0
biSizeImage: 50
biXPelsPerMeter: 2834
biYPelsPerMeter: 2834
biClrUsed: 0
biClrImportant: 0
```

```
*****
BITMAP_FILE_HEADER
=====
bfType: BM
bfSize: 322854
bfReserved1: 0
bfReserved2: 0
bfOffBits: 54
```

```
BITMAP_INFO_HEADER
=====
biSize: 40
biWidth: 400
biHeight: 269
biPlanes: 1
biBitCount: 24
biCompression: 0
biSizeImage: 322800
biXPelsPerMeter: 0
biYPelsPerMeter: 0
```

```
biClrUsed: 0
biClrImportant: 0
```

Λειτουργία 2: Φίλτρο Grayscale

```
$./bmpSteganography -grayscale image1.bmp [ image2.bmp image3.bmp ...]
```

Το όρισμα `-grayscale` εφαρμόζει το **φίλτρο grayscale** το οποίο **μετατρέπει το χρώμα του κάθε pixel σε αποχρώσεις του γκριζου**. Για να γίνει αυτό κατορθωτό θα πρέπει κάθε pixel να μετατρέπεται με την ακόλουθη λογική: **Κάθε pixel καταλαμβάνει 3 bytes** (εφόσον χρησιμοποιείται 24bit βάθος χρώματος). Ειδικότερα, τα bytes αυτά κατανέμονται ως ακολούθως: red=1 byte, green=1 byte, blue=1 byte.

Τα bytes αυτά πολλαπλασιάζονται χρησιμοποιώντας την NTSC (National Television System Committee) εξίσωση: **$0.299 \cdot \text{red} + 0.587 \cdot \text{green} + 0.114 \cdot \text{blue}$** . Το **στρογγυλοποιημένο** αποτέλεσμα της NTSC εξίσωσης πάνω σε κάθε επί μέρους 3-byte RGB ακολουθία του αρχείου θα ονομάζεται απόλυτη φωτεινότητα (luminance). Για παράδειγμα για ένα pixel RGB=(9,90,160), το **luminance είναι $0.299 \cdot 9 + 0.587 \cdot 90 + 0.114 \cdot 160 = 73,761$** . Συνεπώς, η **γκρίζα έκδοση του pixel πρέπει να είναι (74,74,74)**. Εάν επαναλάβουμε την πιο πάνω εξίσωση για όλα τα pixel μιας εικόνας θα έχουμε το ακόλουθο αποτέλεσμα.



Λειτουργία 3: Κωδικοποίηση μιας μυστικής εικόνας μέσα σε μια άλλη εικόνα

```
$./bmpSteganography -encodeStegano nbBits coverImage.bmp secretImage.bmp
```

Το **αποτέλεσμα της εκτέλεσης αυτής της λειτουργίας είναι η δημιουργία μιας εικόνας με το όνομα `new-coverImage.bmp`** η οποία **είναι η εικόνα κάλυμμα με την μυστική εικόνα μέσα της**. Το **όνομα της δημιουργείται από το όνομα της εικόνας κάλυμμα προσθέτοντας το πρόθεμα "new-"**. Η παράμετρος **`nbBits` δείχνει τον αριθμό των bits της εικόνας κάλυμμα που αλλάζουν εισάγοντας άλλα τόσα bits από την μυστική εικόνα**. Παρακάτω φαίνεται ένα παράδειγμα στεγανογραφίας μιας εικόνας:

```
$./bmpSteganography -encodeStegano 4 IMG_6865.bmp IMG_6875.bmp
```

IMG_6865.bmp

IMG_6875.bmp

new-IMG_6865.bmp



Λειτουργία 4: Αποκωδικοποίηση/Επανάκτηση μιας μυστικής εικόνας από μια άλλη εικόνα

```
$ ./bmpSteganography -decodeStegano nbBits encryptedImage.bmp
```

Το αποτέλεσμα της εκτέλεσης αυτής της λειτουργίας είναι η δημιουργία μιας εικόνας με το όνομα `new-encryptedImage.bmp` η οποία είναι η μυστική εικόνα. Το όνομα της δημιουργείται από το όνομα της εικόνας κάλυμμα προσθέτοντας το πρόθεμα "new-". Η παράμετρος `nbBits` δείχνει τον αριθμό των bits της εικόνας κάλυμμα που αλλάζουν εισάγοντας άλλα τόσα bits από την μυστική εικόνα. Παρακάτω φαίνεται ένα παράδειγμα ανάκτησης μιας μυστικής εικόνας από μια άλλη εικόνα:

```
$ ./bmpSteganography -decodeStegano 4 new-IMG_6865.bmp
```

new-IMG_6865.bmp



new-new-IMG_6865.bmp



Λειτουργία 5: Κωδικοποίηση ενός κειμένου μέσα σε μια εικόνα

```
$ ./bmpSteganography -encodeText coverImage.bmp inputText.txt
```

Το αποτέλεσμα της εκτέλεσης αυτής της λειτουργίας είναι η δημιουργία μιας εικόνας με το όνομα `new-coverImage.bmp` η οποία είναι μια εικόνα bmp με ενσωματωμένο μέσα της το κείμενο του αρχείου `inputText.txt`. Το όνομα της δημιουργείται από το όνομα της εικόνας κάλυμμα προσθέτοντας το πρόθεμα "new-". Παρακάτω φαίνεται ένα παράδειγμα στεγανογραφίας ενός κειμένου μέσα σε μια εικόνα:

```
$./bmpSteganography -encodeText tux-bonaparte.bmp poem.txt
```

tux-bonaparte.bmp**poem.txt**

```
PARIS DE NUIT
Jacques Prévert
Trois allumettes une à une allumées dans la nuit
La première pour voir ton visage tout entier
La seconde pour voir tes yeux
La dernière pour voir ta bouche
Et l'obscurité tout entière pour me rappeler tout cela
En te serrant dans mes bras.
```

new-tux-bonaparte.bmp

Λειτουργία 6: Αποκωδικοποίηση/Επανάκτηση μυστικού κειμένου από μια εικόνα

```
$./bmpSteganography -decodeText encryptedImage.bmp msgLength output.txt
```

Το αποτέλεσμα της εκτέλεσης αυτής της λειτουργίας είναι η δημιουργία ενός αρχείου κειμένου με το μήνυμα που έχει ανακτηθεί από την κωδικοποιημένη εικόνα `encryptedImage.bmp`. Για να υλοποιηθεί αυτή η λειτουργία, πρέπει να κάνετε την αντίθετη διαδικασία που κάνατε για την λειτουργία 6. Για να πάρετε το τελικό κείμενο, πρέπει να γνωρίζετε το μήκος του σε χαρακτήρες, το οποίο δίνεται στη γραμμή εντολής από την παράμετρο `msgLength`. Το μήνυμα πρέπει να είναι το ίδιο ως προς τη μορφή του (ίδιες γραμμές) με το μήνυμα που κωδικοποιήθηκε στην λειτουργία 6. Παρακάτω φαίνεται ένα παράδειγμα ανάκτησης ενός κειμένου από μια εικόνα:

```
$./bmpSteganography -decodeText new-tux-bonaparte.bmp 280 new-poem.txt
```

new-tux-bonaparte.bmp**new-poem.txt**

```
PARIS DE NUIT
Jacques Prévert
Trois allumettes une à une allumées dans la nuit
La première pour voir ton visage tout entier
La seconde pour voir tes yeux
La dernière pour voir ta bouche
Et l'obscurité tout entière pour me rappeler tout cela
En te serrant dans mes bras.
```

Λειτουργία 7: Δημιουργία εικόνας bmp από ένα κείμενο

```
$./bmpSteganography -stringToImage sampleImage.bmp inputText.txt
```

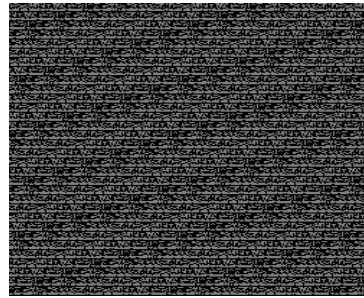
Αυτή η λειτουργία υλοποιεί όσα περιγράφονται στην ενότητα Γ. του κεφαλαίου IV. Το κείμενο του αρχείου `inputText.txt` «μετατρέπεται» σε μια εικόνα `sampleImage.bmp` η οποία θα είναι μια γκριζα εικόνα. Σημειώστε ότι το μήκος του κειμένου πρέπει να είναι αρκετό ώστε να γεμίζει τις διαστάσεις της εικόνας και να γίνει ορατό ένα αποτέλεσμα και με γυμνό μάτι. Η εικόνα `sampleImage.bmp` χρησιμοποιείται μόνο για να αναθέσουμε στην εικόνα που δημιουργείται, τις ίδιες τιμές για τα `BITMAP_FILE_HEADER` και `BITMAP_INFO_HEADER`. Εναλλακτικά θα μπορούσαμε να αγνοήσουμε το `sampleImage.bmp` και να φτιάξουμε μόνοι μας τα `BITMAP_FILE_HEADER` και `BITMAP_INFO_HEADER`. Παρακάτω φαίνεται ένα παράδειγμα:

```
$./bmpSteganography -stringToImage tux-pirate.bmp strFile.txt
```

tux-pirate.bmp



new-tux-pirate.bmp



Λειτουργία 8: Αποκωδικοποίηση/Επανάκτηση μυστικού κειμένου από μια εικόνα bmp

```
$ ./bmpSteganography -imageToString encryptedTextImage.bmp
```

Δημιουργείται ένα αρχείο κειμένου με το όνομα `outputText.txt`

VI. Εξέταση Εργασίας

Κατά την διάρκεια του τελευταίου εργαστηρίου θα γίνει η εξέταση της εργασίας (εάν το επιτρέπουν οι συνθήκες!!!) κατά το οποίο θα πρέπει να γίνει η επίδειξη της σχεδίασης και υλοποίησης της βιβλιοθήκης και του πελάτη σας απ' όλα τα μέλη της ομάδας. Τυχούσα παράληψη παρουσίασης της εργασίας ενδέχεται να συνεπάγεται τον μηδενισμό της εργασίας. Στοιχεία τα οποία ληφθούν υπόψη στην αξιολόγηση της εργασίας σας περιλαμβάνουν: ορθότητα λειτουργίας, στοιχεία επίδοσης: π.χ., ελαχιστοποίηση του χρόνου απόκρισης σε αιτήσεις, το οποίο ορίζεται ως το διάστημα μεταξύ της χρονικής στιγμής που γίνεται η αίτηση και της στιγμής που διεκπεραιώνεται μια λειτουργία, σχεδίαση μονάδων λογισμικού και της βιβλιοθήκης ευρύτερα.

VII. Υποβολή Εργασίας

Για την υποβολή της εργασίας ακολουθήστε τις οδηγίες υποβολής των προηγούμενων εργασιών. Το όνομα του αρχείου zip να είναι το όνομα της ομάδας σας. Μόνο ένα μέλος της ομάδας να κάνει την υποβολή.

Κριτήρια αξιολόγησης:

Υλοποίηση Πελάτη	5
Λειτουργία 1: Εκτύπωση list	5
Λειτουργία 2: Φίλτρο grayscale	10
Λειτουργία 3: encodeStegano	15
Λειτουργία 4: decodeStegano	10
Λειτουργία 5: encodeText	15
Λειτουργία 6: decodeText	10
Λειτουργία 7: stringToImage	10
Λειτουργία 8: imageToString	10
Γενική εικόνα (στοιχισμένος και ευανάγνωστος κώδικας, εύστοχα ονόματα μεταβλητών, σταθερών και συναρτήσεων, σχολιασμός)	5
GPL Προοίμια, Σχεδίαση Προγράμματος, Χρήση SVN, gprof report, valgrind report	5
ΣΥΝΟΛΟ	100
(Προαιρετικό) Επιπλέον λειτουργίες οι οποίες τυγχάνουν να έχουν ενδιαφέρον, πολυπλοκότητα και σκοπό (π.χ., , κτλ.)	10

Καλή Επιτυχία !