Image Processing - HW4

Ημερομηνία παράδοσης: Παρασκευή 9 Ιουνίου

Στην παρούσα εργασία καλείστε να γράψετε ένα πρόγραμμα επεξεργασίας εικόνας. Μπορείτε να φανταστείτε μία εικόνα σαν ένα διδιάστατο πίνακα από εικονοστοιχεία (pixels), μεγέθους όσο και το μέγεθος της εικόνας. Για παράδειγμα, μία εικόνα 712x512 pixels αντιστοιχεί σε ένα δισδιάστατο πίνακα από pixels 712 στηλών και 512 γραμμών (συνηθίζουμε να αναφέρουμε πρώτα το πλάτος και μετά το ύψος της εικόνας).

Οι μέθοδοι και οι κατασκευαστές που προδιαγράφονται στην παρούσα εργασία είναι οι ελάχιστες. Μπορείτε να προσδιορίσετε και επιπλέον μεθόδους ή κατασκευαστές εάν το κρίνετε απαραίτητο.

Η εικόνα τύπου RGB

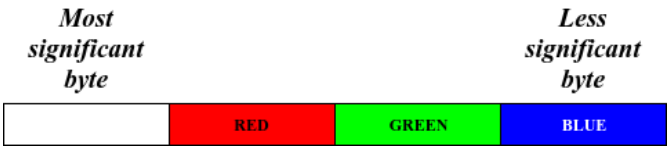
Στον τύπο εικόνας RGB, κάθε εικονοστοιχείο απεικονίζει τις τιμές φωτεινότητας για τα τρία βασικά χρώματα κόκκινο, πράσινο και μπλε, (γνωστά και ως RGB από τα αρχικά των λέξεων red, green, και blue). Θεωρούμε ότι οι τιμές της φωτεινότητας είναι φυσικοί αριθμοί από 0 έως και MAX_ LUMINOCITY (στις εικόνες αυτής της εργασίας η τιμή MAX_ LUMINOCITY είναι 255).

Παραδείγματα RGB τιμών είναι τα εξής:

1. Ένα κόκκινο pixel έχει RGB τιμές 255, 0, 0 (μέγιστη τιμή φωτεινότητας 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 φωτεινότητα για όλα τα χρώματα)
4. Ένα μαύρο pixel προκύπτει από τις RGB τιμές 0, 0, 0 (ελάχιστη φωτεινότητα για όλα τα χρώματα)

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

Η πληροφορία των τριών χρωμάτων μπορεί να αποθηκεύεται με τους εξής δύο εναλλακτικούς τρόπους:



- Σε ένα private πεδίο τύπου int. Ένα παράδειγμα τέτοιας αποθήκευσης δίνεται στο παραπάνω σχήμα.
- Σε τρία πεδία τύπου unsigned char.

Η εικόνα τύπου Grayscale

Στον τύπο εικόνας Grayscale, κάθε εικονοστοιχείο απεικονίζει με μία τιμή φωτεινότητας. Η τιμή της φωτεινότητας είναι φυσικός αριθμός στο διάστημα [0, MAX_LUMINOCITY] (και σε αυτή την περίπτωση η τιμή MAX_ LUMINOCITY είναι 255).

Παραδείγματα Grayscale τιμών είναι τα εξής:

1. Ένα μαύρο pixel προκύπτει από την ελάχιστη τιμή φωτεινότητας 0.
2. Ένα pixel με τιμή φωτεινότητας 128 είναι γκρι.
3. Ένα άσπρο pixel προκύπτει από τη μέγιστη τιμή φωτεινότητας 255.

Σε μία εικόνα αυτού του τύπου υπάρχουν τόσες αποχρώσεις του γκρι, όσες και οι διαφορετικές τιμές της φωτεινότητας, κυμαινόμενες από μαύρο (πολύ χαμηλές τιμές φωτεινότητας) έως και λευκό (πολύ υψηλές τιμές φωτεινότητας).

Αρκεί ένα πεδίο `unsigned char` για να αποθηκεύσει την τιμή της φωτεινότητας ενός ασπρόμαυρου pixel.

Η abstract κλάση Pixel και οι απόγονοι της

Ορίστε την abstract κλάση του Pixel η οποία κατ' ελάχιστο δεν περιέχει καμία μέθοδο ή πεδίο (μπορείτε να προσθέσετε αν το κρίνετε απαραίτητο).

Η κλάση GSCPixel

Η κλάση είναι απόγονος της κλάσης Pixel και περιέχει κατ' ελάχιστον τις παρακάτω μεθόδους.

```
class GSCPixel : public Pixel {
public:
  GSCPixel() = default;
  GSCPixel(const GSCPixel& p);
  GSCPixel(unsigned char value);
  unsigned char getValue();
  void setValue(unsigned char value);
};
```

Η κλάση RGBPixel

Η κλάση είναι απόγονος της κλάσης Pixel και περιέχει κατ' ελάχιστον τις παρακάτω μεθόδους. Ο τρόπος αποθήκευσης των τριών χρωμάτων επιλέγεται από σε εσάς στο/α κατάλληλο/α private πεδίο/α.

```
class RGBPixel : public Pixel {
public:
  RGBPixel() = default;
  RGBPixel(const RGBPixel& p);
  RGBPixel(unsigned char r, unsigned char g, unsigned char b);
  unsigned char getRed() const;
  unsigned char getGreen() const;
  unsigned char getBlue() const;
  void setRed(int r);
  void setGreen(int g);
  void setBlue(int b);
};
```

Η abstract κλάση Image και οι απόγονοι της

Η παρούσα εργασία ορίζει την pure absract κλάση Image η οποία περιγράφεται παρακάτω:

```
class Image {
protected:
  int width;
  int height;
  int max_luminocity;
public:
  int getWidth() const { return width; }
  int getHeight() const { return height; }
```

```
  int getMaxLuminocity() const { return max_luminocity;  }
  void setWidth(int width) { this->width = width; }
  void setHeight(int height) { this->height = height; }
  void setMaxLuminocity(int lum) { this->max_luminocity = lum; }

  virtual Image& operator += (int times) = 0;
  virtual Image& operator *= (double factor) = 0;
  virtual Image& operator !() = 0;
  virtual Image& operator ~() = 0;
  virtual Image& operator *() = 0;
  virtual Pixel& getPixel(int row, int col) const = 0;
  friend std::ostream& operator << (std::ostream& out, Image& image);
};
```

Η επιλογή των κατασκευαστών και του καταστροφέα της κλάσης αφήνεται σε εσάς. Συνιστάται ο καταστροφέας να δηλωθεί ως **virtual**.

Η κλάση περιέχει τις εξής μεθόδους μεταβολής της εικόνας μέσω τελεστών υπερφόρτωσης. Όλες οι μέθοδοι μεταβολής της εικόνας ορίζονται ως **virtual**. Γιατί;

1. **virtual Image& operator += (int times):** Η μέθοδος περιστρέφει την εικόνα δεξιόστροφα τόσες φορές όσες αναφέρεται στην παράμετρο **times**. Αν η παράμετρος είναι αρνητικός αριθμός η εικόνα περιστρέφεται αριστερόστροφα, τόσες φορές όσες ορίζει η απόλυτη τιμή της παραμέτρου **times**.
2. **virtual Image& operator *= (double factor) :** Η μέθοδος μετασχηματίζει το μέγεθος της εικόνας κατά την παράμετρο **factor**. Οι νέες διαστάσεις της εικόνας (πλάτος και ύψος) προκύπτουν εάν πολλαπλασιάσουμε τις αρχικές διαστάσεις με την παράμετρο **factor**. Η μέθοδος επιστρέφει μία αναφορά στο τρέχον αντικείμενο.
3. **virtual Image& operator !() :** Η μέθοδος μετασχηματίζει την εικόνα, ώστε αυτή να έχει αντεστραμμένα τα χρώματα της ή την τιμή της φωτεινότητας αν πρόκειται για ασπρόμαυρη εικόνα. Η μέθοδος επιστρέφει μία αναφορά στο τρέχον αντικείμενο.
    - Η αντιστροφή των χρωμάτων προκύπτει εάν από τη μέγιστη φωτεινότητα αφαιρέσουμε την τρέχουσα φωτεινότητα για κάθε απόχρωση ενός έγχρωμου pixel.
    - Αντίστοιχα, η αντιστροφή φωτεινότητας για ένα ασπρόμαυρο pixel προκύπτει εάν από τη μέγιστη φωτεινότητα αφαιρέσουμε την τρέχουσα φωτεινότητα.
4. **virtual Image& operator ~() :** Η μέθοδος μετασχηματίζει την εικόνα, ώστε αυτή να έχει κατά το δυνατόν ισορροπημένο ιστόγραμμα. Δείτε παρακάτω την περιγραφή της διαδικασίας εξισορρόπησης ιστογράμματος. Η μέθοδος επιστρέφει μία αναφορά στο τρέχον αντικείμενο.
5. **virtual Image& operator *() :** Η μέθοδος μετασχηματίζει την εικόνα, ώστε αυτή να αντιστραφεί κατά τον κάθετο άξονα. Η μέθοδος επιστρέφει μία αναφορά στο τρέχον αντικείμενο.
6. **virtual Pixel& getPixel(int row, int col):** Επιστρέφει μία αναφορά στο pixel που βρίσκεται στη γραμμή row και τη στήλη col της εικόνας. Το pixel μπορεί να είναι έγχρωμο ή ασπρόμαυρο ανάλογα με το είδος της εικόνας στην οποία εφαρμόζεται.
7. **friend std::ostream& operator << (std::ostream& out, Image& image):** Φιλική συνάρτηση η οποία εκτυπώνει τα περιεχόμενα της εικόνας κατά το πρότυπο NetBPM. Εάν πρόκειται για ασπρόμαυρη εικόνα την εκτυπώνει σε μορφή PGM, ενώ εάν πρόκειται για έγχρωμη εικόνα την εκτυπώνει σε μορφή PPM. Οι εκτυπώσεις έχουν τη μορφή κειμένου και όχι δυαδικού αρχείου, δηλαδή οι ασπρόμαυρες εικόνες έχουν magic number **P2** και οι έγχρωμες **P3**.

Η ασπρόμαυρη εικόνα

Γράψτε την κλάση της ασπρόμαυρης εικόνας η οποία έχει κατ' ελάχιστο τους παρακάτω κατασκευαστές και μεθόδους. Η κλάση περιέχει ένα πεδίο **`GSCPixel*`** ή **`GSCPixel**`** μέσω του οποίου αποθηκεύεται το σύνολο των εικονοστοιχείων της εικόνας.

```
class GSCImage : public Image {
public:
  GSCImage();
  GSCImage(const GSCImage& img);
  GSCImage(const RGBImage& grayscaled);
  GSCImage(std::istream& stream);
  ~GSCImage();

  GSCImage& operator = (const GSCImage& img)

  virtual Image& operator += (int ) override ;

  virtual Image& operator *= (double factor) override;
  virtual Image& operator !  () override;
  virtual Image& operator ~  () override;
  virtual Image& operator *  () override;

  virtual Pixel& getPixel(int row, int col) const override;

  friend std::ostream& operator << (std::ostream& out, Image& image) ;
};
```

Η κλάση υλοποιεί τις μεθόδους που περιγράφονται στην κλάση **`Image`**. Επιπλέον, έχει τους εξής κατασκευαστές και τον ανάλογο καταστροφέα.

1. **`GSCImage()`**: Κατασκευαστής χωρίς ορίσματα. Αρχικοποιεί κατάλληλα τα πεδία της κλάσης.
2. **`GSCImage(const GSCImage& img)`**: Κατασκευαστής αντιγραφέας.
3. **`GSCImage(const RGBImage& grayscaled)`**: Κατασκευαστής που δημιουργεί μία ασπρόμαυρη εικόνα από μία έγχρωμη εικόνα. Ο μαθηματικός τύπος μετατροπής ενός εικονοστοιχείου από έγχρωμο σε ασπρόμαυρο είναι **`Gray = Red * 0.3 + Green * 0.59 + Blue * 0.11`**.
4. **`GSCImage(std::istream& stream)`**: Κατασκευαστής που δημιουργεί μία εικόνα διαβάζοντας από το stream εισόδου που δίνεται ως παράμετρος. Ο κατασκευαστής ξεκινάει να διαβάζει εφόσον έχουμε διαβάσει στην αρχή του αρχείο του magic number με τιμή **`P2`**.

Η κλάση διαθέτει τις επιπλέον μεθόδους:

1. **`GSCImage& operator = (const GSCImage& img)`**: Η μέθοδος αναθέτει το περιεχόμενο του δεξιού τελεστέου στον αριστερό τελεστέο. Επιστρέφει μία αναφορά στο τρέχον αντικείμενο.

Τέλος, ορίζεται ως φιλική, η συνάρτηση υπερφόρτωσης του τελεστή **`<<`**, η οποία περιγράφηκε στην κλάση `Image`.

Η έγχρωμη εικόνα

Γράψτε την κλάση της έγχρωμης εικόνας η οποία έχει κατ' ελάχιστο τους παρακάτω κατασκευαστές και μεθόδους. Η κλάση περιέχει ένα πεδίο **`RGBPixel*`** ή **`RGBPixel**`** μέσω του οποίου αποθηκεύεται το σύνολο των εικονοστοιχείων της εικόνας.

```
class RGBImage : public Image {
public:
  RGBImage();
  RGBImage(const RGBImage& img);
  RGBImage(std::istream& stream);
  ~RGBImage();

  RGBImage& operator = (const RGBImage& img);

  virtual Image& operator += (int ) override ;
  RGBImage operator + (int ) ;

  virtual Image& operator *= (double factor) override;
  virtual Image& operator !() override;
  virtual Image& operator ~() override;
  virtual Image& operator *() override;

  virtual Pixel& getPixel(int row, int col) const override;
  friend std::ostream& operator << (std::ostream& out, Image& image);
};
```

Η κλάση υλοποιεί τις μεθόδους που περιγράφονται στην κλάση **`Image`**. Επιπλέον, έχει τους εξής κατασκευαστές και τον ανάλογο καταστροφέα.

1. **`RGBImage()`**: Κατασκευαστής χωρίς ορίσματα. Αρχικοποιεί κατάλληλα τα πεδία της κλάσης.
2. **`RGBImage(const RGBImage& img)`**: Κατασκευαστής αντιγραφέας.
3. **`RGBImage(std::istream& stream)`**: Κατασκευαστής που δημιουργεί μία εικόνα διαβάζοντας από το stream εισόδου που δίνεται ως παράμετρος. Ο κατασκευαστής ξεκινάει να διαβάζει εφόσον έχουμε διαβάσει στην αρχή του αρχείο το magic number με τιμή **`P3`**.

Η κλάση διαθέτει τις επιπλέον μεθόδους:

1. **`RGBImage& operator = (const RGBImage& img)`**: Η μέθοδος αναθέτει το περιεχόμενο του δεξιού τελεστέου στον αριστερό τελεστέο. Επιστρέφει μία αναφορά στο τρέχον αντικείμενο.

Τέλος, ορίζεται ως φιλική, η συνάρτηση υπερφόρτωσης του τελεστή **<<**, η οποία περιγράφηκε στην κλάση `Image`.

Περιγραφή των αλγορίθμων επεξεργασίας της εικόνας

Μεταβολή του μεγέθους της εικόνας

Μπορούμε να μεταβάλλουμε το μέγεθος της εικόνας για τιμές της παραμέτρου factor στο συνεχές διάστημα **`(0, 2]`** ως εξής:

Για κάθε εικονοστοιχείο `[row, col]` της νέας εικόνας υπολογίζουμε τα τέσσερα εικονοστοιχεία της αρχικής εικόνας, τα οποία συμμετέχουν στη διαμόρφωση του χρώματος ως εξής:

`P11 = [r1,c1], P12 = [r1, c2], P21= [r2, c1], P22 = [r2,c2]`, όπου τα **`r1, r2, c1, c2`** υπολογίζονται ως εξής:

- `r1 = min( floor(row/factor), height-1)`
- `r2 = min( ceil(row/factor), height-1)`
- `c1 = min( floor(col/factor), width-1)`
- `c2 = min( ceil(col/factor), width-1)`

Η φωτεινότητα του νέου pixel προκύπτει ως ο μέσος όρος της φωτεινότητας των τεσσάρων pixel που υπολογίστηκαν παραπάνω.

Εξισορρόπηση ιστογράμματος

Τι είναι το ιστόγραμμα μιας εικόνας

Για να κατανοήσετε την έννοια του ιστογράμματος υποθέστε ότι έχετε μία ασπρόμαυρη εικόνα και ένα πίνακα εύρους όσο και η μέγιστη φωτεινότητα + 1, (από 0 έως MAX_LUMINOCITY). Το ιστόγραμμα δημιουργείται εάν στην τυχαία θέση **k** του πίνακα τοποθετήσουμε τον αριθμό των εικονοστοιχείων της εικόνας με τιμή φωτεινότητας **k**. Επομένως, το ιστόγραμμα αποτελεί την κατανομή της φωτεινότητας των εικονοστοιχείων της εικόνας.

Ιστόγραμμα έγχρωμων εικόνων

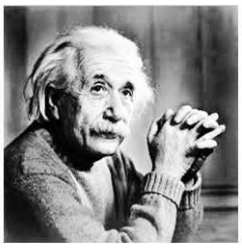
Σε εικόνες τύπου RGB το ιστόγραμμα είναι δύσκολο να οριστεί, καθώς και οι τρεις τιμές RGB συμβάλλουν στη φωτεινότητα. Για τον λόγο αυτό προτιμάμε τη μετατροπή της εικόνας σε YUV και τον υπολογισμό του ιστογράμματος πάνω στην παράμετρο Y που αποτελεί τον μέτρο της φωτεινότητας της εικόνας. Δείτε παρακάτω τις εξισώσεις μετατροπής ενός RGB pixel σε YUV και αντίστροφα.

Για τις ασπρόμαυρες εικόνες δεν είναι αναγκαία η μετατροπή τους σε YUV. Αρκεί να δουλέψουμε με την τιμή φωτεινότητας κάθε εικονοστοιχείου.

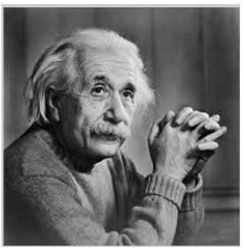
Εξισορρόπησης ιστογράμματος

Συχνά μία εικόνα εμφανίζεται σκοτεινή και συγκεκριμένα τμήματα της είναι δυσδιάκριτα. Η ευκρίνεια της εικόνας μπορεί να βελτιωθεί σημαντικά εάν μεγαλώσουμε το εύρος των τιμών φωτεινότητας της εικόνας. Ενδεικτικό είναι το παρακάτω σχήμα.

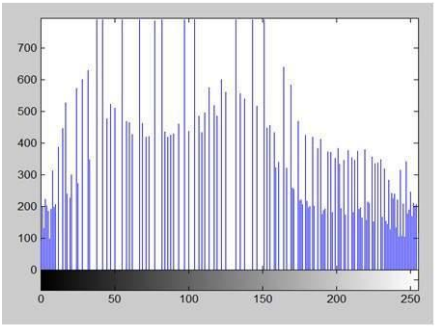
New Image



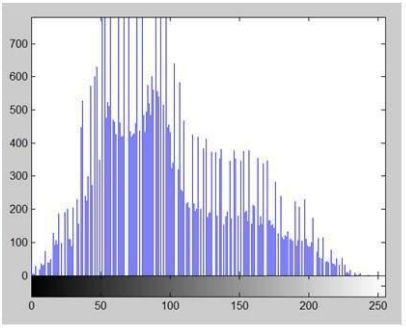
Old image



New Histogram



Old Histogram



Η διαδικασία της εξισορρόπησης του ιστογράμματος έχει ως εξής:

1. Υπολογίζουμε το ιστόγραμμα. Πριν τον υπολογισμό του ιστογράμματος μην παραλείψετε να μηδενίσετε όλες τις τιμές του σχετικού πίνακα.
2. Με βάση το ιστόγραμμα, υπολογίζουμε την κατανομή πιθανότητας της φωτεινότητας των εικονοστοιχείων της εικόνας. Η πιθανότητα ένα εικονοστοιχείο να έχει φωτεινότητα **k** ορίζεται ως το πηλίκο του αριθμού των εικονοστοιχείων με πιθανότητα **k** προς το σύνολο των εικονοστοιχείων της εικόνας.
3. Με βάση το βήμα 2 υπολογίζουμε την αθροιστική κατανομή πιθανότητας, δηλαδή την πιθανότητα ένα εικονοστοιχείο να έχει τιμή φωτεινότητας μικρότερη ή ίση της τιμής **k**, που υπολογίσαμε παραπάνω. Η τιμή υπολογίστηκε αποθηκεύεται στη θέση **k** του πίνακα αθροιστικής πιθανότητας.
4. Επιλέγουμε τη μέγιστη τιμή φωτεινότητας την οποία θέλουμε να έχει η νέα εικόνα. Στο σύστημα YUV, η μέγιστη τιμή φωτεινότητας είναι 235 (δοκιμάστε στο λογιστικό φύλλο να βάλετε τιμές RGB 255, 255, 255 και δείτε ποια είναι η τιμή της παραμέτρου Y). Επιλέξτε ως μέγιστη τιμή φωτεινότητας την τιμή 235 τόσο για τις έγχρωμες όσο και τις για τις ασπρόμαυρες εικόνες. Τις ασπρόμαυρες εικόνες θα πρέπει να τις διαχειριστείτε ως έγχρωμες, δηλαδή κάθε pixel της εικόνας να το μετατρέψετε σε RGB, όπου και τα τρία χρώματα θα έχουν την ίδια τιμή φωτεινότητας και στη συνέχεια σε YUV. Αφού εξισορροπήσετε την εικόνα θα πρέπει να μετατρέψετε κάθε YUV pixel πίσω σε RGB και στη συνέχεια σε ασπρόμαυρο, ακολουθώντας την αντίστροφη διαδικασία.
5. Πολλαπλασιάζουμε τον πίνακα που προκύπτει από το βήμα 3 (αθροιστική κατανομή πιθανότητας) με τη μέγιστη τιμή φωτεινότητας του βήματος 4 και αποθηκεύουμε την τιμή σε έναν πίνακα ακεραίων αποκόπτοντας με αυτόν τον τρόπο το δεκαδικό μέρος του αριθμού που προκύπτει από το γινόμενο.
6. Από τον βήμα 5 προκύπτει η προτεινόμενη μεταβολής της φωτεινότητας. Ας υποθέσουμε ότι στη θέση X του πίνακα ακεραίων που προκύπτει στο βήμα 5 είναι αποθηκευμένη η τιμή Y. Με βάση το παραπάνω, όλα τα εικονοστοιχεία με φωτεινότητα X θα πρέπει να μετατραπούν σε εικονοστοιχεία με φωτεινότητα Y. Μετασχηματίζοντας τη φωτεινότητα για όλα τα

εικονοστοιχεία της εικόνας λαμβάνουμε την τελική μετασχηματισμένη εικόνα η οποία είναι εξισορροπημένη ως προς το ιστόγραμμα της.
7. Η εικόνα μετατρέπεται από YUV σε RGB ή GSC ανάλογα με το είδος της αρχικής εικόνας.

Παρατήρηση: Η εξισορρόπηση ιστογράμματος δε μεταβάλλει τη χρώματα της εικόνας, παρά μόνο τη φωτεινότητα τους.

Αναλυτικά παραδείγματα για την εξισορρόπηση ιστογράμματος μπορείτε να δείτε εδώ και εδώ.

Το κυρίως πρόγραμμα

Η μέθοδος main του κυρίως προγράμματος θα πρέπει να βρίσκεται στο αρχείο **`hw4.cpp`** και το τελικό εκτελέσιμο θα έχει όνομα **`hw4`**. Για τη μεταγλώττιση του κώδικα σας καλείστε να παρέχετε κατάλληλο **`Makefile`**, το οποίο θα μεταγλωττίζει όλα τα αρχεία υποχρεωτικά με το flag **`-fsanitize=address`**.

Το κυρίως πρόγραμμα διαβάζει έγχρωμες ή ασπρόμαυρες εικόνες PPM ή PGM και τις αποθηκεύει σε μία δομή της επιλογής σας, που λειτουργεί ως βάση δεδομένων, η οποία αντιστοιχεί μοναδικά ονόματα σε εικόνες. Κάθε νέα εικόνα που διαβάζεται συνδέεται με ένα μοναδικό όνομα (token) το οποίο υποχρεωτικά ξεκινάει με το χαρακτήρα **`'$'`**. Ο τελικός χρήστης μπορεί να κάνει τα εξής:

- να εισάγει φωτογραφίες συνδυάζοντας τες με διαφορετικό token κάθε μία.
- να διαγράψει ένα token μαζί με τη φωτογραφία στην οποία αντιστοιχεί.
- να εφαρμόσει διαφορετικές ή και τις ίδιες πράξεις επεξεργασίας εικόνας, σειριακά, πάνω στην ίδια φωτογραφία. Για παράδειγμα, ο τελικός χρήστης μπορεί να περιστρέψει, να αλλάξει το μέγεθος και να εξισορροπήσει το ιστόγραμμα μια εικόνας πριν την εξάγει και πάλι σε αρχείο.
- Οποιοδήποτε φωτογραφία έχει εισαχθεί, μπορεί να εξαχθεί πίσω στο filesystem, αποθηκεύοντας τη με διαφορετικό όνομα ή σε άλλο κατάλογο του filesystem. Η φωτογραφία που εξάγεται μπορεί να έχει υποστεί επεξεργασία ή όχι.

Περιορισμοί:

- Κάθε token καταχωρείται ως το μοναδικό αναγνωριστικό ενός δείκτη τύπου **`Image (Image*)`**. Ενδεικτικά, η κλάση **`Token`** δίνεται παρακάτω. Απαγορεύεται να διατηρείτε άλλη συνοδευτική μετα-πληροφορία για κάθε εικόνα, πέρα από τα δύο πεδία που δίνονται εδώ (π.χ. την επιπλέον πληροφορία αν η εικόνα είναι έγχρωμη ή ασπρόμαυρη).

```
class Token {
private:
  string name;
  Image* ptr;
public:
  Token(const string& ="", Image* = nullptr);
  ~Token();
  string getName() const;
  Image* getPtr() const;
  void setName(const string& );
  void setPtr(Image* ptr);
};
```

- Ο αριθμός των διαφορετικών μοναδικών αναγνωριστικών που μπορούν να καταχωρηθούν από το πρόγραμμα είναι θεωρητικά άπειρος. Τα αναγνωριστικά πρέπει να καταχωρούνται σε λίστα ή δυναμικό πίνακα (vector) της επιλογής σας.
- Νέα αναγνωριστικά token μπορούν να εισαχθούν σε συνδυασμό με εικόνες ή να διαγραφούν μαζί με τις συνοδές τους εικόνες. Δεν υπάρχουν περιορισμοί σχετικά με το μέγεθος του κάθε μοναδικού ονόματος, αρκεί να ξεκινά με τον χαρακτήρα **`'$'`**.

Το κυρίως πρόγραμμα που θα γράψετε κάνει τα εξής. Σε μία ατέρμονη επανάληψη διαβάζει γραμμή - γραμμή και αναλύει τα περιεχόμενα της γραμμής που διάβασε σύμφωνα με τις παρακάτω εντολές.

Σε σχέση με τις εντολές και τα μηνύματα που ακολουθούν ισχύουν οι ακόλουθες συμβάσεις.

- Σε όλα τα μηνύματα που εκτυπώνονται, εκτυπώνεται και χαρακτήρας αλλαγής γραμμής μετά από αυτά.
- Στις εντολές που περιγράφονται παρακάτω οι παράμετροι που εισάγει ο χρήστης περικλείονται μέσα σε **<>**. Ο χρήστης δεν εισάγει τους χαρακτήρες **<** και **>**, αλλά εισάγονται εδώ προς διευκρίνηση του τι είναι παράμετρος και τι δεσμευμένη/σταθερή λέξη. Σε όλες τις εντολές που ακολουθούν, οι δεσμευμένες/σταθερές λέξεις κάθε εντολής εμφανίζονται χωρίς **<>**.

Σε όλες τις εντολές ελέγχονται με την ακόλουθη σειρά τα εξής:

- Εφόσον υπάρχουν δεσμευμένες/σταθερές λέξεις, ελέγχεται εάν οι λέξεις αυτές εμφανίζονται στη σωστή θέση μέσα στην εντολή. Επιπλέον, ελέγχεται εάν το **`token`** που εισάγει ο χρήστης ξεκινάει με τον χαρακτήρα **`'$'`**. Εάν παραβιάζεται οποιαδήποτε από τα παραπάνω εμφανίζεται το μήνυμα **`"\n-- Invalid command! --"`**.
- Σε όλες τις εντολές με εξαίρεση τις εντολές **`import`** και **`quit`**, ελέγχεται εάν το **`$token`** που δίνεται είναι καταχωρημένο ήδη στη βάση. Εάν δεν είναι καταχωρημένο εκτυπώνεται το μήνυμα **`"[ERROR] Token $tokenId not found!"`**.

Εντολή - `import`	**Περιγραφή**
`i <filename> as <$token>`	Εισαγωγή ενός αρχείου εικόνας με όνομα **`filename`** από το filesystem, που αντιστοιχεί στο μοναδικό αναγνωριστικό **`$token`**.

- Εάν το αρχείο **`filename`** δεν υπάρχει ή ο χρήστης δεν έχει δικαίωμα να το ανοίξει για διάβασμα εκτυπώνει το μήνυμα **`"[ERROR] Unable to open filename"`**, όπου **`filename`** το μονοπάτι που προσδιόρισε ο χρήστης.
- Εάν το αρχείο δεν έχει το σωστό μοναδικό αναγνωριστικό magic number (P2 ή P3) εκτυπώνει το μήνυμα **`"[ERROR] Invalid file format"`**.
- Εάν η ανάγνωση ήταν επιτυχής, αλλά υπάρχει άλλο **`$token`** ενεργό με το ίδιο όνομα εκτυπώνει το μήνυμα **`"[ERROR] Token $token exists"`**.
- Εάν η ανάγνωση ήταν επιτυχής και η καταχώρηση το αρχείου ήταν επίσης επιτυχής εκτυπώνει το μήνυμα **`"[OK] Import $token"`**.

Για την ανάγνωση από αρχείο θα χρησιμοποιηθεί υποχρεωτικά η παρακάτω συνάρτηση (αντιγράψτε τη με το χέρι, μην κάνετε copy - paste):

```
Image* readNetpbmImage(const char* filename) {
  ifstream f(filename);
```

```
  if(!f.is_open()) {
    std::cout << "[ERROR] Unable to open " << filename << std::endl;
  }
  Image* img_ptr = nullptr;
  string type;

  if(f.good() && !f.eof())
    f >> type;
  if(!type.compare("P3")) {
    img_ptr = new RGBImage(f);
  }
  else if(!type.compare("P2")) {
    img_ptr = new GSCImage(f);
  }
  else if(f.is_open()) {
    std::cout << "[ERROR] Invalid file format" << std::endl;
  }
  return img_ptr;
}
```

Εντολή - `export`	**Περιγραφή**
`e <$token> as <filename>`	Εξαγωγή της εικόνας που είναι συνδυασμένη με το **`$token`** σε αρχείο με μονοπάτι **`filename`**. Εάν η εικόνα είναι ασπρόμαυρη εξάγεται σε μορφή PGM, ενώ εάν η εικόνα είναι έγχρωμη εξάγεται σε μορφή PPM.
• Εάν το αρχείο **`filename`** υπάρχει ήδη εκτυπώνει το μήνυμα **`"[ERROR] File exists"`**. • Εάν το αρχείο **`filename`** δεν υπάρχει, αλλά ο χρήστης δεν έχει δικαίωμα να το δημιουργήσει και να γράψει σε αυτό εκτυπώνεται το μήνυμα **`"[ERROR] Unable to create file"`**. • Εάν η εξαγωγή ήταν επιτυχής εκτυπώνει **`"[OK] Export $token"`**,όπου **`$token`** είναι το μοναδικό αναγνωριστικό που προσδιόρισε ο χρήστης.	

Εντολή - `delete`	**Περιγραφή**
`d <$token>`	Διαγράφει το μοναδικό αναγνωριστικό **`$token`** από τη μνήμη μαζί με την εικόνα που αντιστοιχεί σε αυτό.
Εφόσον διαγραφεί με επιτυχία το **`$token`** εκτυπώνεται το μήνυμα **`"[OK] Delete $token"`**, όπου **`$token`** είναι το μοναδικό αναγνωριστικό που προσδιόρισε ο χρήστης.	

Εντολή - `color inversion`	**Περιγραφή**
`n <$token>`	Αντιστρέφει τη φωτεινότητα της εικόνας που αντιστοιχεί στο μοναδικό αναγνωριστικό $token.
Εφόσον αντιστραφεί η φωτεινότητα της εικόνας με επιτυχία εκτυπώνεται **`"[OK] Color`**	

Inversion $token"**, όπου **$token είναι το μοναδικό αναγνωριστικό που προσδιόρισε ο χρήστης.

Εντολή - equalize	**Περιγραφή**
z <$token>	Γίνεται εξισορρόπηση ιστογράμματος της εικόνας που αντιστοιχεί στο μοναδικό αναγνωριστικό **$token**.
Εφόσον εξισορροπηθεί η φωτεινότητα της εικόνας με επιτυχία εκτυπώνεται **"[OK] Equalize $token"**, όπου **$token** είναι το μοναδικό αναγνωριστικό που έδωσε ο χρήστης.	

Εντολή - mirror	**Περιγραφή**
m <$token>	Γίνεται αντιστροφή (mirror) κατά τον κάθετο άξονα της εικόνας που αντιστοιχεί στο μοναδικό αναγνωριστικό **$token**.
Εφόσον αντιστραφεί η εικόνα με επιτυχία εκτυπώνεται **"[OK] Mirror $token"**, όπου **$token** είναι το μοναδικό αναγνωριστικό που έδωσε ο χρήστης.	

Εντολή - grayscale	**Περιγραφή**
g <$token>	Εάν η εικόνα είναι ασπρόμαυρη δε γίνεται καμία ενέργεια. Εάν η αρχική εικόνα είναι έγχρωμη, από την έγχρωμη εικόνα παράγεται η αντίστοιχη ασπρόμαυρη, η οποία συνδέεται με το ίδιο αναγνωριστικό **$token**. Η προηγούμενη έγχρωμη εικόνα διαγράφεται.
• Εάν η εικόνα μετασχηματιστεί με επιτυχία εκτυπώνεται **"[OK] Grayscale $token"**, όπου **$token** είναι το μοναδικό αναγνωριστικό που έδωσε ο χρήστης. • Εάν αρχική εικόνα ήταν ήδη ασπρόμαυρη εκτυπώνεται **"[NOP] Already grayscale $token"** , όπου **$token** είναι το μοναδικό αναγνωριστικό που έδωσε ο χρήστης.	

Εντολή - scale	**Περιγραφή**
s <$token> by <factor>	Μεταβάλλεται το μέγεθος της εικόνας που αντιστοιχεί στο μοναδικό αναγνωριστικό **$token**. Οι νέες διαστάσεις προκύπτουν από τις αρχικές εάν αυτές πολλαπλασιαστούν με τον αριθμό κινητής υποδιαστολής **factor**.
Εάν μετασχηματιστεί με επιτυχία το μέγεθος της εικόνας εκτυπώνεται **"[OK] Scale $token"**, όπου **$token** είναι το μοναδικό αναγνωριστικό που έδωσε ο χρήστης.	

Εντολή - rotate	**Περιγραφή**

`r <$token> clockwise <X> times`	Περιστρέφεται η εικόνα που αντιστοιχεί στο μοναδικό αναγνωριστικό **`$token`** δεξιόστροφα τόσες φορές όσες περιγράφει η ακέραια παράμετρος **`X`**. Εάν το **`X`** είναι αρνητικός αριθμός η εικόνα περιστρέφεται αριστερόστροφα τόσες φορές όσες περιγράφει η απόλυτη τιμή του **`X`**.
Εάν περιστραφεί με επιτυχία η εικόνα εκτυπώνεται "**`[OK] Rotate $token`**", όπου **`$token`** είναι το μοναδικό αναγνωριστικό που έδωσε ο χρήστης.	

Εντολή - `quit`	**Περιγραφή**
`q`	Τερματίζει το πρόγραμμα. Πριν τον τερματισμό πρέπει να ελευθερωθεί όλη η μνήμη που προηγουμένως δεσμεύτηκε.

Οδηγίες Αποστολής

Η αποστολή της εργασίας θα γίνει μέσω της πλατφόρμας autolab. Υπάρχει μέγιστο όριο 30 δωρεάν υποβολών που μπορείτε να κάνετε. Μετά από αυτό τον αριθμό σας αφαιρείται ένας πόντος για κάθε επιπλέον υποβολή.

Για την υποβολή ακολουθήστε τα εξής βήματα:

- Δημιουργήστε ένα κατάλογο με όνομα **hw4**, μέσα στον οποίο θα αντιγράψετε όλα τα αρχεία με κατάληξη .cpp, .hpp, καθώς και το Makefile το οποίο μεταγλωττίζει την εργασία σας. Υπενθυμίζεται ότι το **`Makefile`** πρέπει να μεταγλωττίζει όλα τα αρχεία με το επιπλέον flag **`-fsanitize=address`**.
- Συμπιέστε τον κατάλογο **hw4** σε μορφή zip. Μέσα στο zip πρέπει να υπάρχει και ο κατάλογος **hw4**. Το αρχείο που προκύπτει πρέπει να έχει όνομα **hw4.zip.**
- Συνδέεστε στο autolab και επιλέγετε το μάθημα ECE326_2023 (S23) και από αυτό την εργασία HW4.
- Για να υποβάλετε την εργασία σας κάνετε click στην επιλογή “I affirm that I have compiled with this course academic integrity policy…” και πατάτε submit. Στη συνέχεια επιλέγετε το αρχείο hw2.zip που δημιουργήσατε παραπάνω.

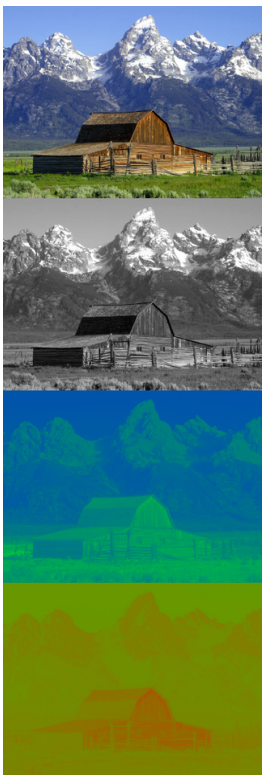
Παράρτημα Α - Η εικόνα YUV

Μία εικόνα **YUV** είναι μία εικόνα της οποίας κάθε pixel περιγράφεται από τις τρεις τιμές ακέραιες τιμές Y, U και V. Η τιμή Y αντιπροσωπεύει την τιμή της φωτεινότητας για το pixel και οι τιμές U, V αντιπροσωπεύουν πληροφορία χρώματος.

Στην διπλανή εικόνα, δίνεται το παράδειγμα μιας έγχρωμης εικόνας και πως αυτή διακρίνεται στις παραμέτρους Y, U και V, οι οποίες ακολουθούν με τη σειρά αμέσως μετά την κορυφαία εικόνα.

Παρατηρήστε ότι η εικόνα Y (2η στη σειρά εικόνα) είναι ασπρόμαυρη καθώς περιγράφει μόνον τη φωτεινότητα κάθε εικονοστοιχείου.

*πηγή εικόνας Wikipedia



Μετατροπή εικονοστοιχείων από RGB σε YUV και αντίστροφα

Υπάρχει η δυνατότητα μετατροπής ενός pixel από RGB σε YUV και αντίστροφα. Οι συναρτήσεις μετατροπής ενός RGB pixel σε YUV δίνονται παρακάτω:

Y = ((66 * R + 129 * G + 25 * B + 128) >> 8) + 16
U = ((-38 * R - 74 * G + 112 * B + 128) >> 8) + 128
V = ((112 * R - 94 * G - 18 * B + 128) >> 8) + 128

Αντίστοιχα, οι συναρτήσεις μετατροπής ενός YUV pixel σε RGB είναι οι εξής:

C = Y - 16
D = U - 128
E = V - 128

R = clip((298 * C   + 409 * E + 128) >> 8)
G = clip((298 * C - 100 * D - 208 * E + 128) >> 8)
B = clip((298 * C + 516 * D   + 128) >> 8)

Η συνάρτηση **clip** κάνει το εξής: εάν η τιμή εισόδου της είναι αρνητική την κάνει 0, ενώ εάν είναι μεγαλύτερη από 255 την κάνει 255.

Σημείωση: Κατά τη μετατροπή YUV σε RGB μπορείτε να υποθέσετε με ασφάλεια ότι η μέγιστη τιμή φωτεινότητας της RGB εικόνας είναι 255.

Μπορείτε να **κατεβάσετε εδώ ένα λογιστικό φύλλο** που μετατρέπει τιμές RGB σε YUV και αντίστροφα.

Παράρτημα Β - Format PPM και PGM για έγχρωμες και ασπρόμαυρες εικόνες

Στην παρούσα εργασία θα χρησιμοποιήσετε για διάβασμα από αρχείο και αποθήκευση σε αρχείο ασπρόμαυρες ή έγχρωμες εικόνες που είναι αποθηκευμένες **σε μορφή κειμένου** και τα αρχεία τους έχουν συνήθως κατάληξη .PGM (PGM example) ή **.PPM** (PPM example). Η μορφή ενός αρχείου PGM ή PPM έχει ως εξής:

1. Ξεκινάει με το αλφαριθμητικό **P2** για ασπρόμαυρες εικόνες PGM ή **P3** για έγχρωμες εικόνες PPM.
2. Ακολουθεί ένας ακέραιος που αντιστοιχεί στο πλάτος της εικόνας (σε εικονοστοιχεία).
3. Ακολουθεί ένας ακέραιος που αντιστοιχεί στο ύψος της εικόνας (σε εικονοστοιχεία).
4. Ακολουθεί ένας ακέραιος που αντιστοιχεί στη μέγιστη τιμή φωτεινότητας της εικόνας.

Τα σημεία 1-4 αποτελούν την κεφαλίδα του αρχείου. Στη συνέχεια

- **Ασπρόμαυρες εικόνες:** Για κάθε εικονοστοιχείο μία ασπρόμαυρης εικόνας εμφανίζεται ένας ακέραιος αριθμός.
- **Έγχρωμες εικόνες:** Για κάθε εικονοστοιχείο μία έγχρωμης εικόνας εμφανίζονται 3 ακέραιοι για τα τρία RGB χρώματα κόκκινο, πράσινο και μπλε με τη σειρά.

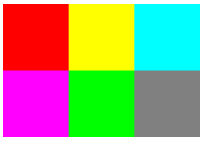
Κάθε αρχείο ξεκινά με την πληροφορία του επάνω αριστερού εικονοστοιχείου (θεωρούμε ως πρώτη σειρά εικονοστοιχείων, την κορυφαία σειρά), έπεται το αμέσως δεξιότερο εικονοστοιχείο, μέχρι να φτάσουμε στο δεξιότερο στοιχείο της κορυφαίας σειράς. Στη συνέχεια το αρχείο συνεχίζει από τον αριστερότερο στοιχείο της επόμενης σειράς κ.ο.κ.

Για παράδειγμα, εάν έχουμε μία έγχρωμη εικόνα μεγέθους **5x4** εικονοστοιχείων στο αρχείο θα έχουμε αποθηκευμένους **5x4x3** ακεραίους που αντιστοιχούν στην πληροφορία των εικονοστοιχείων της εικόνας. Είναι προφανές ότι η τιμή κάθε ακεραίου δεν πρέπει να υπερβαίνει τη μέγιστη τιμή φωτεινότητας που ορίστηκε στην αρχή του αρχείου (στο σημείο 4).

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 (whitespace χαρακτήρες).

Παρακάτω δίνονται δύο παραδείγματα εικόνων PPM και PGM μεγέθους **3x2** και δίπλα η εικόνα που αντιστοιχεί σε αυτές, σε μεγέθυνση.

```
P3
3 2
255
255 0 0 255 255 0 0 255 255
255 0 255 0 255 0 128 128 128
```



```
P2
3 2 255
0 255 128
55 200 128
```

Ένα αρχείο PPM ή PGM μπορείτε να το δείτε σε Linux από οποιοδήποτε πρόγραμμα προβολής ή επεξεργασίας εικόνων (π.χ. gwenview, gimp). Σε Windows μπορείτε να το δείτε μέσω gimp ή μέσω

του προγράμματος OpenSeeIt (δεν απαιτεί εγκατάσταση). Εναλλακτικά, μπορείτε να βλέπετε τις φωτογραφίες και online εδώ.