Προγραμματισμός Υπολογιστών με C++

Εργασία Ακαδημαϊκού Έτους 2019-20

Ημερομηνία Παράδοσης Εργασίας: **12 Ιανουαρίου 2020**, ομάδες 1-2 ατόμων

1. Περίληψη

Στην εργασία αυτή θα φτιάξετε μια εφαρμογή επεξεργασίας εικόνας. Θα ξεκινήσετε από τη βασική υλοποίηση μιας γενικής κλάσης αναπαράστασης δεδομένων δύο διαστάσεων, την οποία στη συνέχεια θα χρησιμοποιήσετε για να ορίσετε και υλοποιήσετε μια εξειδικευμένη κλάση για την αναπαράσταση, το άνοιγμα και την αποθήκευση μιας εικόνας με χρήση του μορφότυπου PPM. Για το τελευταίο, θα χρειαστεί να υλοποιήσετε 2 συναρτήσεις για την ανάγνωση και εγγραφή δεδομένων από και προς αρχεία τύπου PPM, τις οποίες θα ενσωματώσετε σε μια βιβλιοθήκη. Τη βιβλιοθήκη θα την χρησιμοποιήσετε από την εφαρμογή σας για να υλοποιηθεί η λειτουργικότητα (αντίστοιχες μέθοδοι) της κλάσης της εικόνας. Αφού ολοκληρωθεί η λειτουργικότητα αναπαράστασης και αποθήκευσης φορτώματος εικόνας, θα δηλώσετε και υλοποιήσετε μια σειρά από φίλτρα επεξεργασίας εικόνας τα οποία θα μπορείτε να εφαρμόζετε σε μια εικόνα μέσω της εφαρμογής σας δηλώνοντάς τα παραμετρικά ως ορίσματα της εφαρμογής στη γραμμή εκτέλεσης. Ακολουθεί λεπτομερής περιγραφή της εργασίας και των βημάτων της.

2. Δεδομένα

Σας δίνεται ένα σύνολο από αρχεία κεφαλίδας τα οποία εξηγείται παρακάτω πώς χρησιμοποιούνται. Τα αρχεία αυτά **ΔΕΝ ΤΑ ΤΡΟΠΟΠΟΙΕΙΤΕ**, παρά τα χρησιμοποιείτε για να ορίσετε τις συναρτήσεις και μεθόδους τους και να υλοποιήσετε την κύρια εφαρμογή και τις κλάσεις που αυτή απαιτεί. Ακόμα, σας δίνεται ένα λογισμικό για την ανάγνωση εικόνων σε μορφότυπο αρχείου PPM. Εναλλακτικά, μπορείτε να κάνετε προεπισκόπηση των εικόνων με οποιαδήποτε εφαρμογή υποστηρίζει αυτό τον μορφότυπο (π.χ. το ελεύθερο λογισμικό επεξεργασίας εικόνας GIMP). Τέλος, σας δίνεται ένα σύνολο εικόνων ελέγχου για να πειραματιστείτε.

3. Υλοποίηση

3.1. Βιβλιοθήκη ppm

Να δημιουργηθεί μια εξωτερική βιβλιοθήκη με όνομα ppm.lib (ή libppm.a για χρήστες του gcc) που να αντιστοιχεί στην υλοποίηση των συναρτήσεων ReadPPM και WritePPM που ορίζονται στο ppm.h που σας δίνεται. Ο ρόλος της ReadPPM είναι να διαβάζει από ένα αρχείο τα δεδομένα της αντίστοιχης εικόνας και να παράγει έναν buffer δεδομένων εικόνας (πλάτος X ύψος τριάδες από float δεδομένα). Τα δεδομένα αυτά τα χρησιμοποιείτε εσείς στη συνέχεια για να αρχικοποιήσετε κατάλληλα ένα αντικείμενο τύπου Image. Η WritePPM παίρνει σαν όρισμα μια υπάρχουσα εικόνα σε μορφή πίνακα δεδομένων όμοιο με αυτόν που επιστρέφει η ReadPPM, και γράφει τα δεδομένα σε ένα αρχείο σύμφωνα με το μορφώτυπο PPM. **Οδηγίες για την εσωτερική δομή του μορφότυπου PPM Θα βρείτε στην ενότητα 4**.

Η βιβλιοθήκη θα πρέπει να είναι ένα ανεξάρτητο Project μέσα στο Solution της εργασίας και να παράγει στατική βιβλιοθήκη. Για τη debug έκδοση της εφαρμογής σας, να παράγεται αντίστοιχα η

βιβλιοθήκη ppm_d.lib (ή libppm_d.a για gcc). Η κατάλληλη έκδοση της βιβλιοθήκης θα πρέπει αργότερα να ενσωματωθεί στην εφαρμογή σας και να χρησιμοποιηθούν οι 2 υλοποιημένες συναρτήσεις μέσα σε αυτή, ως έχουν. Για τη διαδικασία σύνδεσης και ενσωμάτωσης κώδικα από στατικές βιβλιοθήκες, δείτε τις σχετικές διαφάνειες του μαθήματος.

Δεν ξεχνάμε ότι ενώ στο μορφότυπο PPM οι τιμές έντασης ανά κανάλι χρώματος (Red, Green Blue) είναι αποθηκευμένες σε ακέραιη μορφή με εύρος 0-255, οι τιμές που θα πρέπει να αποθηκεύονται στους πίνακες δεδομένων της εικόνας που επιστρέφει η ReadPPM και παίρνει ως είσοδο η WritePPM είναι αριθμοί δεκαδικοί ανάλογα προσαρμοσμένοι στο διάστημα [0,1].

Για να ελέγξετε τη σωστή λειτουργία της βιβλιοθήκης σας, προτείνεται να δημιουργήσετε ένα απλό πρόγραμμα ως ανεξάρτητο Project μέσα στο Solution σας, το οποίο ανοίγει ένα αρχείο PPM, διαβάζει τα δεδομένα του σε έναν πίνακα και τα ξανα-αποθηκεύει σε ένα αρχείο τύπου PPM με άλλο όνομα αρχείου.

3.2. Η κλάση Array2D

Για την υλοποίηση τόσο της κλάσης Image που θα παριστάνει τα δεδομένα και τις πράξεις μιας εικόνας στην εφαρμογή μας, όσο και για την υλοποίηση των φίλτρων εικόνας παρακάτω, σας δίνεται η δήλωση της templated κλάσης Array2D, της οποίας θα πρέπει να δώσετε τον ορισμό των μεθόδων της (αρχείο array2d.h). Για το σκοπό αυτό, να δημιουργήσετε ένα αρχείο array2d.hpp το οποίο ενσωματώνεται ήδη από το αρχείο κεφαλίδας array2d.h και μέσα σε αυτό να περάσετε τους ορισμούς των μεθόδων. Πληροφορίες για τη λειτουργικότητα των μεθόδων και τις παραμέτρους τους θα βρείτε στο αρχείο κεφαλίδας, στα σχόλια πάνω από κάθε μέθοδο.

Μπορείτε να ελέγξετε αν θέλετε την καλή λειτουργία της κλάσης σας δημιουργώντας ένα ανεξάρτητο εκτελέσιμο (Project) μέσα στο Solution σας, όπου φτιάχνετε απλά αντικείμενα τύπου Array2D<float> καλώντας κατασκευαστές και άλλες μεθόδους για να διαπιστώσετε ότι δουλεύουν σωστά.

3.3. Η κλάση Image

Δηλώστε την κλάση Image ως κοινό απόγονο της κλάσης Array<T>, με τον τύπο T να είναι math::Vec3<float> [1]. Η κλάση Vec3 ορίζεται στο αρχείο vec3.h. Η κλάση Image έχει, επιπλέον των μεθόδων της Array<T>, τις μεθόδους load() και save(), τις οποίες κληρονομεί από την αφηρημένη κλάση ImageIO (αρχείο imageio.h) με πολλαπλή κληρονομικότητα. Οι δύο αυτές μέθοδοι πρέπει να χρησιμοποιούν τις συναρτήσεις ReadPPM και WritePPM της βιβλιοθήκης ppm.lib που φτιάξατε.

Στην κλάση Image, το κάθε εικονοστοιχείο παριστάνεται με ένα στιγμιότυπο της Vec3 αποθηκευμένο στη γραμμική δομή buffer που κληρονομείται από το Array2D. Άρα ο υλοποιημένος και κληρονομημένος τελεστής () της Array2D, στην απόγονη κλάση Image μας επιστρέφει ένα Vec3 για το εικονοστοιχείο (x,y) της εικόνας από το οποίο μπορούμε να δούμε τις τρεις συνιστώσες χρώματος ως τις συντεταγμένες του διανύσματος.

[1] Για ευκολία, μπορείτε να δημιουργήσετε ένα ψευδώνυμο Color για τον τύπο math::Vec3<float> και να χρησιμοποιείτε αυτό στη δήλωση και υλοποίηση της Image και της Filter.

Για να ελέγξετε την ορθότητα του κώδικά σας, μια πρώτη δοκιμή θα μπορούσε να είναι η αντικατάσταση του κώδικά επαλήθευσης που φτιάξατε στην ενότητα 3.1 με δημιουργία ενός στιγμιότυπου Image και χρήση των μεθόδων load και save αντί για την κλήση των συναρτήσεων ReadPPM και WritePPM.

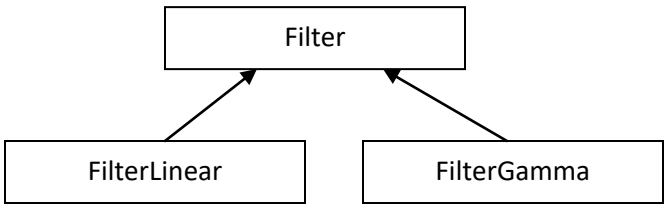
3.4. Φίλτρα Εικόνας

Δηλώστε και υλοποιήστε μια βασική κλάση Filter της οποίας ο ρόλος είναι να υλοποιεί ένα φίλτρο επεξεργασίας εικόνας. Η μόνη μέθοδος που διαθέτει η κλάση Filter είναι η pure virtual μέθοδος:

```
Image operator << (const Image & image)
```

Η μέθοδος αυτή επενεργεί πάνω σε μια εικόνα (βλ. παρακάτω) και να την αλλοιώνει, παράγοντας μια νέα και θα πρέπει να μπορεί να κληθεί πολυμορφικά για τους απογόνους της κλάσης Filter. Οι απόγονοι αυτής της κλάσης θα υλοποιούν διαφορετικά ο κάθε ένας αυτή τη μέθοδο, ανάλογα με το είδος του φίλτρου. Η κλάση θα πρέπει να διαθέτει επίσης τους κατάλληλους constructors (default, copy).

Δηλώστε και υλοποιήστε δύο εξειδικευμένα φίλτρα ως απογόνους της βασικής κλάσης Filter, σύμφωνα με το παρακάτω διάγραμμα κληρονομικότητας:



Η κάθε μία υλοποιεί με διαφορετικό τρόπο τον τελεστή << που κληρονομεί. Συγκεκριμένα, η FilterLinear εφαρμόζει σε κάθε Pixel $\mathbf{p}(x, y)$ της εικόνας εισόδου (δεξιός τελεστέος) τον τύπο:

$$\mathbf{p}'(x, y) = \mathbf{a} \cdot \mathbf{p}(x, y) + \mathbf{c},$$

όπου $\mathbf{p}'(x, y)$ το εικονοστοιχείο στη συντεταγμένη (x,y) της εικόνας που δημιουργεί και επιστρέφει ο τελεστής και **a**, **c** δύο σταθερές τιμές math::Vec3<float>. Για παράδειγμα, αν $\mathbf{a} = (-1, -1, -1)$ και $\mathbf{c} = (1,1,1)$ το φίλτρο θα πρέπει να παράγει το αρνητικό της πρωτότυπης εικόνας. Σημείωση: όλα τα αποτελέσματα της παραπάνω πράξης θα πρέπει να τα περιορίζετε στο διάστημα [0,1] για κάθε κανάλι χρώματος.

Ο τελεστής << της FilterGamma υλοποιεί την πράξη:

$$\mathbf{p}'(x,y) = \mathbf{p}(x,y)^{\gamma},$$

όπου κάθε κανάλι χρώματος ενός εικονοστοιχείου υψώνεται στον εκθέτη *γ*. Τυπικές τιμές για το *γ* μεταξύ του 0.5 και του 2.0.

Οι κλάσεις FilterLinear και FilterGamma πρέπει να διαθέτουν επιπρόσθετα πεδία από τη βασική κλάση για να φυλάνε τις παραμέτρους τους, καθώς και τους αντίστοιχους κατασκευαστές.

3.5. Κύρια Εφαρμογή

Υλοποιήστε μια κύρια εφαρμογή ως ανεξάρτητο Project στο Solution σας, η οποία θα παράγει ένα εκτελέσιμο με όνομα filter.exe για τη Release έκδοση και filterd.exe για τη Debug έκδοση. Για τη δημιουργία του εκτελέσιμου **θα πρέπει να συνδέσετε τη βιβλιοθήκη ppm.lib** (την αντίστοιχη Debug ή Release έκδοσή της) που φτιάξατε.

Η εφαρμογή, αφού φορτώσει μια εικόνα ppm που προσδιορίζουμε από την κονσόλα, θα εφαρμόζει πάνω σε αυτή μια σειρά από φίλτρα επεξεργασίας εικόνας των παραπάνω 2 τύπων. Ο τύπος, ο αριθμός και η σειρά των φίλτρων εικόνας που θα επενεργούν πάνω στην εικόνα προσδιορίζονται κι αυτά από τα ορίσματα της εφαρμογής. Η τροποποιημένη εικόνα θα πρέπει να αποθηκεύεται σε αρχείο ppm κι αυτή, αφού πρώτα αλλαχθεί το όνομα αρχείου προσθέτοντας το πρόθεμα “filtered_” στην αρχή του ονόματος της εικόνας εισόδου.

Ο τρόπος εκτέλεσης του προγράμματος ακολουθεί το παρακάτω μοτίβο:

filter -f [φίλτρο 1] [παράμετροι] … -f [φιλτρο k] [παράμετροι] [όνομα αρχείου εικόνας]

Μετά τον προσδιοριστή –f ακολουθεί το όνομα του φίλτρου, που μπορεί να είναι είτε linear είτε gamma και αμέσως μετά μια σειρά από ορίσματα που αντιστοιχούν στις εξειδικευμένες παραμέτρους του κάθε φίλτρου. Το φίλτρο linear θα πρέπει να ακολουθείται από 6 αριθμούς float που αντιστοιχούν στις συντεταγμένες των και **a**, **c** διανυσμάτων. Το φίλτρο gamma ακολουθείται από 1 παράμετρο που αντιστοιχεί στον εκθέτη *γ*. Συνοπτικά, η παραμετροποίηση φαίνεται στον ακόλουθο πίνακα:

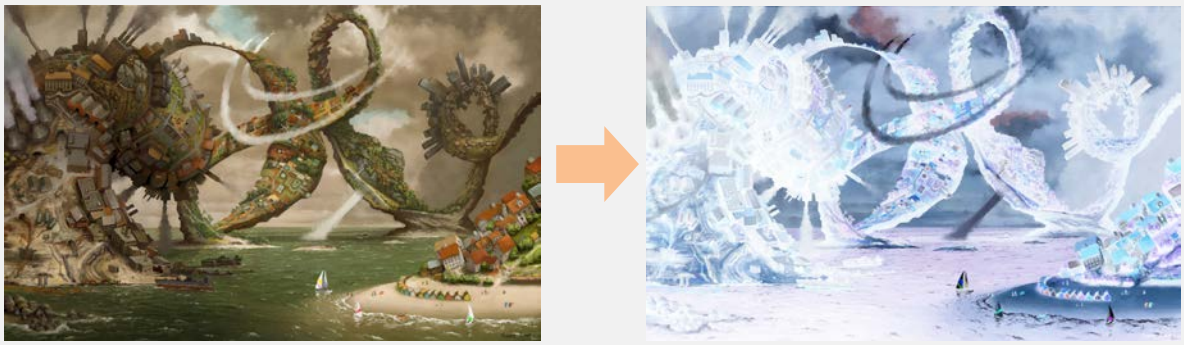
Κλάση που υλοποιεί το φίλτρο	Προσδιοριστής φίλτρου	Παράμετροι
FilterLinear	linear	aR aG aB cR cG cB
FilterGamma	gamma	γ

Το αρχείο εικόνας εισόδου δίνεται πάντα τελευταίο. Αν δε δοθεί φίλτρο, η εφαρμογή πρέπει να τερματίζει χωρίς να κάνει τίποτα. Αν δεν προσδιοριστεί αρχείο ή δε δοθεί το όνομα κάποιου φίλτρου σωστά (ή καθόλου), η εφαρμογή πρέπει να βγάζει σχετικό μήνυμα λάθους και να τερματίζει.

Ο αριθμός των φίλτρων και το είδος τους προσδιορίζεται κατά την εκτέλεση με βάση τις παραμέτρους που παίρνει η εφαρμογή από τη γραμμή εντολών. Επίσης, ένα φίλτρο ενδέχεται να εφαρμόζεται πολλαπλές φορές και με διαφορετικές παραμέτρους κάθε φορά. Η σειρά εφαρμογής των φίλτρων είναι από αριστερά προς δεξιά. Παραδείγματα:

```
> filter -f gamma 2.0 -f linear -1 -1 -1 1 1 1 image01.ppm
```

Εφαρμόζει πρώτα ένα FilterGamma φίλτρο με γ=2, και στη συνέχεια αντιστρέφει την εικόνα (αρνητικό).



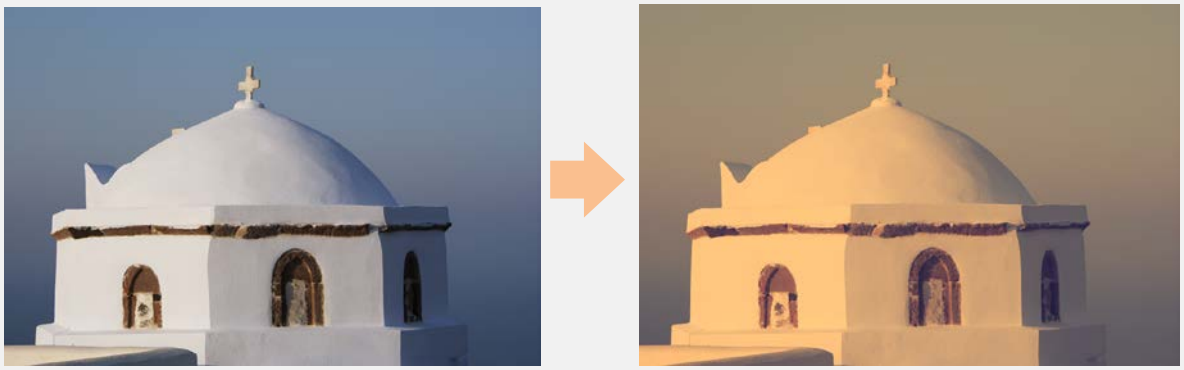
```
> filter -f gamma 2.0 -f gamma 0.5 image01.ppm
```

Εφαρμόζει 2 διαδοχικά φίλτρα FilterGamma τα οποία αλληλοαναιρούνται.

```
> filter -f linear 2 2 2 0 0 0 image01.ppm
```

Διπλασιάζει τη φωτεινότητα της εικόνας. Τιμές πάνω από την τιμή κορεσμού (1.0) αποκόπτονται στο 1.0 (βλ. παραπάνω οδηγίες για την υλοποίηση του FilterLinear << operator).

```
> filter –f gamma 0.7 -f linear 1 0.8 0.3 0.1 0.1 0.3 –f gamma 1.2
```

image02.ppm
Εφαρμόζει 3 φίλτρα που αλλοιώνουν μη ομοιόμορφα τα χρώματα της εικόνας. 

4. Μορφότυπος αρχείων εικόνας ppm

Στην εργασία σας θα χρειαστεί να φορτώσετε ένα αρχείο εικόνας με βάση έναν γνωστό, απλό μορφότυπο, τον PPM, υλοποιώντας τη συνάρτηση LoadPPM, της οποίας η ακριβής δήλωση σας δίνεται.

Για την εργασία θα υποστηρίξουμε την έκδοση P6 του ppm format. Το ppm format είναι ένα μικτό ASCII/binary format το οποίο είναι από τις απλούστερες αναπαραστάσεις αποθήκευσης εικόνων. Τις ακριβείς προδιαγραφές του μπορείτε να τις βρείτε στη διεύθυνση:

http://paulbourke.net/dataformats/ppm/

Παρατίθεται η επεξήγηση στα Ελληνικά εδώ:

Ένα αρχείο τύπου PPM αποτελείται από δύο μέρη, την κεφαλίδα και τα δεδομένα της εικόνας. Η κεφαλίδα απαρτίζεται από τουλάχιστο τρία τμήματα, χωρισμένα υπό φυσιολογικές συνθήκες με το χαρακτήρα της νέας γραμμής (CR/LF – “return”: ‘\n’), ή τουλάχιστο από τον κενό χαρακτήρα (space, tab). Το πρώτο τμήμα (ή η πρώτη γραμμή, αν είναι χωρισμένη η κεφαλίδα με “return”) είναι ο προσδιοριστής του τύπου του ppm format που ακολουθεί, ο οποίος μπορεί να είναι είτε “P3” είτε “P6” (προφανώς χωρίς τα εισαγωγικά!). Εμείς υποστηρίζουμε μόνο το P6 format.

Οι επόμενες δύο πληροφορίες που περιέχονται στην κεφαλίδα είναι το πλάτος και το ύψος της εικόνας σε pixels, δοσμένα ως ASCII κείμενο.

Το τέταρτο τμήμα της κεφαλίδας αναγράφει την μέγιστη επιτρεπόμενη τιμή που μπορεί να αποθηκευτεί σε κάθε pixel. Αν η τιμή είναι μέχρι 255, τότε η εικόνα αποθηκεύει ένα byte για κάθε κανάλι χρώματος (κόκκινο, πράσινο, μπλε) σε κάθε pixel. Αν ο αριθμός που αναγράφεται είναι μεγαλύτερος του 255, τότε χρειάζονται 2 bytes. Εμείς υποστηρίζουμε μόνο 24bit εικόνες, δηλαδή τιμές μέχρι 255.

Επιπροσθέτως, μεταξύ των παραπάνω γραμμών ή στο τέλος μιας γραμμής (δηλαδή μετά την πληροφορία που περιέχει), μπορεί να τοποθετηθεί οπουδήποτε μέσα στην κεφαλίδα ένα σχόλιο. Το σχόλιο ξεκινάει πάντα με το χαρακτήρα '#' και εκτείνεται ως το τέλος της γραμμής (μέχρι το χαρακτήρα "return" – '\n').

Αν κατά την ανάγνωση της κεφαλίδας κάτι πάει στραβά, εμφανίζουμε ένα σχετικό σφάλμα στην κονσόλα στο οποίο και περιγράφουμε ακριβώς το πρόβλημα. Πιθανά σφάλματα είναι (με τη σειρά που μπορεί να προκύψουν): δεν υπάρχει το αρχείο που προσπαθούμε να ανοίξουμε ή δεν ανοίγει, προσπάθεια να διαβάσουμε ppm που δεν είναι τύπου P6, λείπει η οριζόντια ή/και η κάθετη διάσταση της εικόνας, απέτυχε να διαβαστεί η μέγιστη τιμή αποθήκευσης ή είναι μεγαλύτερη του 255.

Σημείωση: Για την απλούστερη υλοποίηση της συνάρτησης ανάγνωσης της εικόνας, να θεωρήσετε ότι δεν υπάρχουν σχόλια (δηλαδή τμήματα κειμένου που ξεκινάνε από το χαρακτήρα '#') και επομένως, η κεφαλίδα περιέχει μόνο τα απαραίτητα πεδία της εικόνας.

Παραδείγματα κεφαλίδων:

`P6 1024 788 255`	`P6`
	`1024`
	`788`
	`255`
		`P6`
		`1024 788`
		`255`

Μετά το τελευταίο πεδίο της κεφαλίδας (τη μέγιστη τιμή δηλαδή) πάντα υπάρχει ένας ακριβώς διαχωριστικός χαρακτήρας μεταξύ της κεφαλίδας και του σώματος των δεδομένων, συνήθως ένα κενό (space) ή '\n'.

Όταν ο τύπος του format είναι P6, τότε τα δεδομένα της εικόνας είναι αποθηκευμένα σε bytes (δυαδικό format), ένα byte για κάθε χρωματική συνιστώσα (R,G,B). Τα pixels σώζονται κατά γραμμές που περιέχουν τριπλέτες RGB από αριστερά προς τα δεξιά με πρώτη γραμμή (γραμμή 0) την πάνω και τελευταία την κάτω.

Προσοχή: **Θα πρέπει να υλοποιήσετε το μηχανισμό ανάγνωσης και εγγραφής του μορφότυπου ppm μόνοι σας** και όχι να ενσωματώσετε κάποια υπάρχουσα βιβλιοθήκη που τον υλοποιεί (προφανώς επίσης θα πρέπει να είστε σε θέση να εξηγήσετε τι κάνει ο κώδικας).

5. Επιπρόσθετη βαθμολογία

5.1. Φίλτρο Εξομάλυνσης (bonus +0.3)

Δηλώστε και υλοποιήστε μια επιπρόσθετη κλάση απόγονο της Filter, την FilterBlur που υλοποιεί τον τελεστή << έτσι ώστε για κάθε pixel (i, j) της εικόνας προορισμού, διαβάζει μια περιοχή NXN pixels της εικόνας εισόδου με κέντρο το (i, j) και θέτει ως χρώμα του pixel (i, j) το βεβαρυμμένο άθροισμα των (το πολύ NXN) έγκυρων pixels της NXN γειτονιάς με βάρος $1/N^2$. Ο υπολογισμός μπορεί εύκολα να προκύψει από τον τύπο:

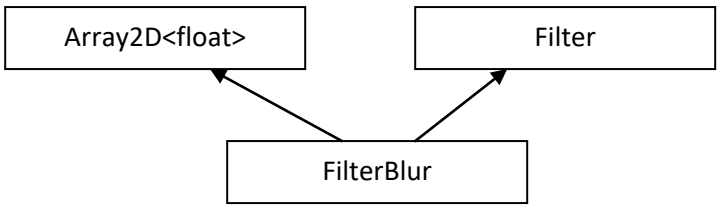
$$\mathbf{p}'(i,j) = \sum_{m=-N/2}^{N/2} \sum_{n=-N/2}^{N/2} \mathbf{p}(i+m, j+n) f(m+\frac{N}{2}, n+\frac{N}{2})$$

Το άθροισμα «τρέχει» για όλα τα εικονοστοιχεία $\mathbf{p}(i+m, j+n)$, για τα οποία $0 \leq i+m < w$, $0 \leq j+n < h$ (εντός ορίων), όπου w, h το πλάτος και ύψος της εικόνας. Πολλαπλασιάζει δε το εικονοστοιχείο με το αντίστοιχο σημείο του πίνακα του φίλτρου f.

Η αντίστοιχη με τα άλλα φίλτρα παραμετροποίηση από τη γραμμή εντολών είναι:

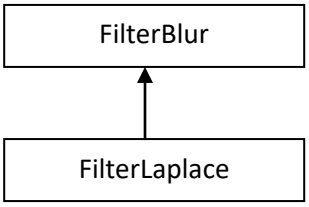
Κλάση που υλοποιεί το φίλτρο	Προσδιοριστής φίλτρου	Παράμετροι
FilterBlur	blur	N (περιττός αριθμός)

Για την υλοποίηση του φίλτρου αυτού, η FilterBlur κληρονομεί τόσο από την Filter, όσο και από την Array2D<float> η οποία και προσφέρει την εσωτερική αναπαράσταση των τιμών του πίνακα *f*. Στο συγκεκριμένο φίλτρο, όλα τα κελιά έχουν την τιμή $1/N^2$ και προσδίδονται στον πίνακα κατά την αρχικοποίηση αντικειμένου τύπου FilterBlur (για συγκεκριμένη παράμετρο κατασκευής N).



5.2. Φίλτρο Εύρεσης Ακμών (bonus +0.3)

Δηλώστε και υλοποιήστε μια επιπρόσθετη κλάση απόγονο της FilterBlur, την FilterLaplace.



Η FilterLaplace που υλοποιεί τον τελεστή << έτσι ώστε για κάθε pixel (i,j) της εικόνας προορισμού, διαβάζει μια περιοχή 3X3 pixels της εικόνας εισόδου με κέντρο το (i,j) και θέτει ως χρώμα του pixel (i,j) της εικόνας εξόδου το βεβαρυμμένο άθροισμα των (το πολύ 3X3) έγκυρων pixels της 3X3 γειτονιάς με βάρος $f(k,l)$, όπου το $f(k,l)$ προκύπτει από τον ακόλουθο πίνακα 3X3 που αντιστοιχεί κελί προς κελί με τη γειτονιά 3X3 των Pixels της εικόνας πάνω στην οποία εφαρμόζεται το φίλτρο:

$$f = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

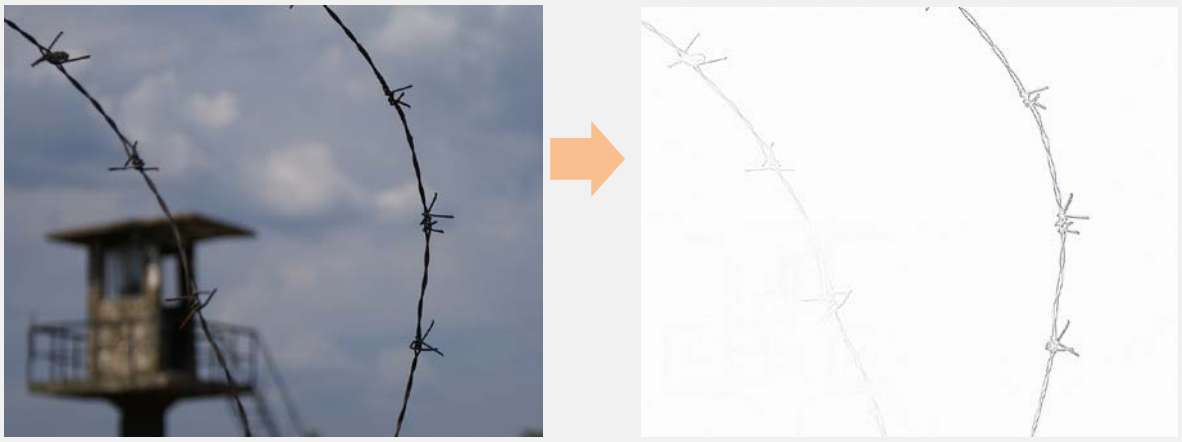
$$\mathbf{p}'(i,j) = \mathbf{max}\left(\mathbf{0}, \mathbf{min}\left(\mathbf{1}, \sum_{m=-1}^{1}\sum_{n=-1}^{1} \mathbf{p}(i+m, j+n) f(m+1, n+1)\right)\right)$$

Για τις πράξεις min και max μπορείτε να χρησιμοποιήσετε τις μεθόδους clampToLowerBound(val) και clampToUpperBound(val) του Vec3.

Στη γραμμή εντολών, δηλώνουμε τη χρήση του φίλτρου Laplace με το λεκτικό laplace και καμία παράμετρο, π.χ.:

```
> filter -f laplace –f linear -2 -2 -2 1 1 1 image01.ppm
```

Βρίσκει τις ακμές της εικόνας, τις ενισχύει Χ2 και τις αντιστρέφει.



6. Γενικές Οδηγίες

Για την υλοποίηση των προγραμμάτων, μπορείτε να χρησιμοποιήσετε όποιο περιβάλλον θέλετε αλλά ο κώδικας που θα παραδώσετε θα πρέπει να είναι σε μορφή Solution του Visual Studio της Microsoft για λειτουργικό Windows 10. Παραδίδετε έτοιμο Visual Studio project και όχι μόνο τα .cpp και .h αρχεία και έχετε επιβεβαιώσει ότι γίνονται σωστά build. Πριν ανεβάσετε την εργασία σας, καθαρίστε τα παραγόμενα από το Visual Studio αρχεία (Build→Clean Solution και διαγραφή του αρχείου .sdf από το solution directory).

Προσοχή: Σε οποιαδήποτε περίπτωση διαπιστωθεί **αντιγραφή** κώδικα ή αδυναμία του εξεταζομένου φοιτητή να **εξηγήσει την υλοποίησή του**, η εργασία **μηδενίζεται** αυτομάτως. Επίσης, το εκτελέσιμο της εργασίας **πρέπει να δουλεύει** για να βαθμολογηθεί η τελευταία.