KTU
COM 2001: Object Oriented Programming

Assignment 4: An Image processing Program

Purpose

The purpose of this assignment is to help you learn or review (1) the fundamentals of the C++ programming language, (2) the inheritance and operator overloading concepts.

Rules

Make sure you do coding of your program on your own. You can ask for help from your friends to understand the problem, figure out how to tackle the problem, but you may never copy someones code and present it as your own. If this is detected, you will receive a caution, your code will not be marked. If you copied your work from a friend of yours, you will both receive a caution and your work will not be marked.

Background

Finding the edges in an image is one of the most fundamental parts computer vision. You are required to code a very simple Object Oriented Program that aims to extract edge information from a grayscale image. Furthermore, you are required to perform some logic operations on the binary edge image by overloading certain operators in the created solution.

The Task

You are required to code a simple image processing application using OOP approach. In this system, you are required to model the input images as 2 dimensional arrays where each array can be instantiated from a class. You may choose to inherit a matrix class to create the image class where you will need to overload certain operator to enable image processing features on this class. For example, the following code loads the gray level (8 bit per pixel) image data from a file, where the first two bytes represent height and width of the image.

```
Image im1,im2;
File1>>im1; //load image one to im1 object
File22>>im2; //load image one to im2 object

/* The two images are needed to be read from the binary files. The
first two bytes store image dimensions as stated above after which
each data stored in the file represents a gray level pixel. The
image can have a size of 255x255. */
```

```
im1.sobel();
im2.sobel();
/*The code above applies sobel filter on the image. You can learn
the details on how to apply a sobel filter to an image from the
Internet.*/

im1.threshold(100);
im2.threshold(100);
/* This code above, creates thresholded image where the result
becomes a binary image. The values are either a 0 or a 1.*/

Image im3;
im3 = im1 + im2;  // 1 OR
im3 = im1 * im2;  // 3 AND
im3 = !im1;   // 4 NOT
/*Apart from the filtering functionality you are required to
overload the operators given above where the operations are
defined as logic operations on binary images.*/

File3<<im3;
/* The results should be printed as a text file where 0's will be
represented by `-` and 1's will be represented by '*'. */
```

Functionality

A typical execution of your program from the shell might look like this:

$>./**processImage file1.bin file2.bin fileout.txt**

Step 5: Create a readme File

Use emacs/vi/gedit to create a text file named readme (not readme.txt, or README, or Readme, etc.) that contains:

- Your name, student ID, and the assignment number.
- A description of whatever help (if any) you received from others while doing the assignment, and the names of any individuals with whom you collaborated, as prescribed by the course Policy web page.
- (Optionally) An indication of how much time you spent doing the assignment.
- (Optionally) Your assessment of the assignment: Did it help you to learn? What did it help you to learn? Do you have any suggestions for improvement? Etc.
- (Optionally) Any information that will help us to grade your work in the most favorable light. In particular you should describe all known bugs.

Descriptions of your code should not be in the readme file. Instead they should be integrated into your code as comments.

Your readme file should be a plain text file. Don't create your readme file using Microsoft Word, Hangul (HWP) or any other word processor.

Step 6: Submit

Your submission should include your **project** file, the files that gcc generated from it, and your readme file. You can do that using moodle platform.

Grading

We will grade your work on two kinds of quality: quality from *the user's* point of view, and quality from *the programmer's* point of view.

From the user's point of view, a program has quality if it behaves as it should.

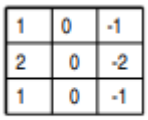
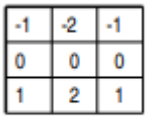
From the programmer's point of view, a program has quality if it is well styled and thereby easy to maintain. For this assignment we will pay particular attention to rules 1-24. These additional rules apply:

- **Names**: You should use a clear and consistent style for variable and function names. One example of such a style is to prefix each variable name with characters that indicate its type. For example, the prefix c might indicate that the variable is of type char, i might indicate int, pc might mean char*, ui might mean unsigned int, etc. But it is fine to use another style -- a style that does not include the type of a variable in its name -- as long as the result is a clear and readable program.
- **Comments**: Each source code file should begin with a comment that includes your name, the number of the assignment, and the name of the file.
- **Comments**: Each function -- especially the main function -- should begin with a comment that describes *what the function does* from the point of view of the caller. (The comment should not describe *how the function works*.) It should do so by *explicitly* referring to the function's parameters and return value. The comment also should state what, if anything, the function reads from the standard input stream or any other stream, and what, if anything, the function writes to the standard output stream, the standard error stream, or any other stream. Finally, the function's comment should state which global variables the function uses or affects. In short, a function's comments should describe the flow of data into and out of the function.
- **Function modularity**: Your program should not consist of one large main function. Instead your program should consist of multiple small functions, each of which performs a single well-defined task.
- **Line lengths**: Limit line lengths in your source code to 72 characters. Doing so allows us to print your work in two columns, thus saving paper.

Special Notes

Edge detection in an image with sobel is performed by applying horizontal and vertical Sobel masks to the image.

Web Source

 h_x vertical Sobel mask	S_x: the result obtained by applying vertical Sobel mask
 h_y horizontal Sobel mask	S_y: the result obtained by applying horizontal Sobel mask
$S = \sqrt{{S_x}^2 + {S_y}^2}$	the final edge image (***S***) will be calculated by this equation

KTU

BIL2001: Nesne Yönelimli Programlama

Ödev 4: Bir Görüntü İşleme Programı

Amaç

Ödevin amacı (1) C++ programlama dilinin temellerini, (2) miras ve "operator overloading" kavramlarını öğrenmeye ve anlamaya yardımcı olmaktır.

Kurallar

Kaynak kodu tek başına kodlamanız beklenmektedir. Problemi anlamak, çözümlemek ve algoritma geliştirmek için arkadaşlarınızdan yardım alabilirsiniz, ancak kaynak kodu başkalarından asla kopyalamayın ve kendi kodunuz gibi sunmayın. Bu tespit edilirse uyarı alırsınız ve kaynak kodunuza 0 puan verilir. Eğer sınıf arkadaşlarınızdan birinin kodunu kopyalarsanız ikiniz de uyarı alırsınız ve ödeviniz 0 puan olarak değerlendirilir.

Arkaplan (Senaryo)

Bilgisayarla Görü uygulamalarında en temel işlemlerden biri kenarların bulunmasıdır. Sizden gri seviye bir görüntüde kenar bilgisini çıkaran Nesne Yönelimli bir program kodlamanız istenmektedir. Ayrıca yazacağınız programda ek olarak istenen operatörleri "overload" yaparak ikili kenar görüntüsünde bazı lojik işlemleri yerine getirmeniz istenmektedir.

İş Tanımı

Sizden Nesne Yönelimli Programlama yaklaşımı kullanarak basit bir görüntü işleme uygulaması kodlamanız istenmektedir. Yazacağınız programda görüntüleri 2 boyutlu dizi şeklinde modellemeniz ve her bir görüntüyü bir sınıfın nesnesi olarak oluşturmanız beklenmektedir. Bunun için matris sınıfından miras yoluyla görüntü sınıfı oluşturabilirsiniz. Görüntü sınıfında istenen işlemleri yerine getirmek için "operator overloading" işlemleri gerçekleştirmeniz gerekmektedir. Örnek olarak, aşağıdaki kod her bir pikseli 8 bit ile ifade edilen gri seviye bir görüntüyü ikili bir dosyadan yüklemektedir. Dosyanın ilk 2 byte bilgisi görüntünün yükseklik ve genişliğini temsil etmektedir.

```
Goruntu g1,g2;
Dosya1>>g1; //load image one to g1 object
Dosya2>>g2; //load image one to g2 object
/* ilk ve ikinci görüntüyü binary dosyalardan oku. Dosyadan
veriler byte seklinde okunacaktır. Dosyadan ilk okunan byte
```

```
görüntünün yüksekliğini ikinci okunan byte ise görüntünün
genisligini, diger okunan her bir byte ise gri seviye görüntünün
parlaklik değerini ifade etmektedir. Görüntünün boyutu en fazla
255x255 olabilir. */

g1.sobel();
g2.sobel();
/* gri seviye goruntuler uzerinde sobel filtresi gezdirme işlemi.
Bu işlemin nasil çalıştığı internetten arastirilarak kolaylıkla
bulunabilir. */

g1.threshold(100);
g2.threshold(100);
/* 100 değerinden az olan pikselleri 0 ve fazla olan pikselleri 1
olarak ayarlayarak goruntu binary formata cevrilir. */

Goruntu g3;
g3 = g1 + g2;  // 1 OR
g3 = g1 * g2;  // 3 AND
g3 = !g1;   // 4 NOT
/* binary görüntüler üzerinde 3 adet ikili görüntü aritmetiği-
"binary image operations" yapilabilmektedir. */

Dosya3<<g3;
/* sonuç görüntü text olarak dosyaya yazılacaktır. Dosyaya 0
değeri yerine `-` ve 1 değeri yerine `*` yazılmalıdır. */
.
```

Kullanım

Yazacağınız programın girdileri 3 adet dosya olacaktır. Tipik bir program çalıştırma komutu ve parametreler aşağıdaki şekilde olabilir.

```
$>./processImage file1.bin file2.bin fileout.txt
```

Kaynak Kod Yazımı ve Gönderimi

- Kaynak kod yazarken kaynak kodun en başına açıklama satırı şeklinde, ad, soyad, numara bilgileri yazılmalıdır.
- Gerekli yerlere açıklama satırları eklenmeli ve kodun okunabilirliğini sağlamak için girintilere ve değişken isimlerine dikkat edilmelidir.
- Ayrıca kaynak kodun görsel açıdan hizalı olmasına dikkat edilmelidir.
- Kodu gönderirken tüm kodları tek bir cpp ya da txt dosyasına yazarak ödev yükleme arayüzüne yüklenmelidir.
- Ödev savunmaları esnasında öğrenciden bu dosyadaki kaynak kodlar kullanılarak çalıştırılabilir duruma getirilmesi ve gerekirse değişiklik yapması istenebilecektir.

Puanlama

Çalışmanız kullanıcı ve programcı olmak üzere iki açıdan değerlendirilecektir.

Kullanıcı açısından değerlendirilirken programın davranışları ve çalışması test edilecektir.

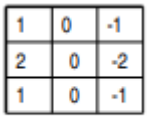
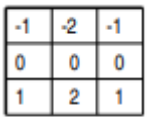
Programcı açısından test edilirken kaynak kodun yazma stili ve bakım kolaylığı göz önüne alınacaktır. Bu açıdan aşağıdaki kurallara uyulması beklenmektedir.

- **İsimlendirme**: Açık ve tutarlı değişken, sınıf ve fonksiyon isimlendirme stili kullanılmalıdır. Örneğin her değişkenden önce türünü ifade eden bir karakter kullanılabilir. char için c, unsigned int için ui kullanılabilir. Ayrıca bakınız: hungarian, camel case, PEP8
- **Açıklamalar**: Her kaynak kod dosya adı, öğrenci ismi, numarası ve dosya düzenleme tarihi (sürüm numarası) gibi bilgileri en başta açıklama satırları şeklinde içermelidir. Her fonksiyonun -- özellikle main() fonksiyonu -- ne iş yaptığı açıklama satırı olarak yazılmalıdır. Parametre ve geri dönüş değerleri hakkında yorum satırları eklenmelidir.
- **Modüler Programlama**: Program çok büyük bir main fonksiyonu içermemelidir. Bunun yerine daha kısa ve belirli bir iş yapan birden çok fonksiyona bölünmelidir. Fonksiyon parametre sayısı makul sayıda olmalıdır. Aynı durum sınıf tasarımında da dikkate alınmalıdır.
- **Satır Uzunlukları**: Kaynak dosyada bir satır uzunluğu 72 karakter ile sınırlı tutulmaldır.
- **OOP**: Kaynak koddaki tasarımda OOP kavramları dikkate alınmalıdır.

Diğer Notlar

Sobel ile kenar algılama işlemi bir görüntüde yatay ve dikey sobel maskelerinin uygulanması ile gerçekleştirilir.

Kaynak

 h_x dikey Sobel maskesi	S_x: dikey Sobel maskesi uygulanarak elde edilen kenar görüntüsü
 h_y yatay Sobel maskesi	S_y: yatay Sobel maskesi uygulanarak elde edilen kenar görüntüsü
$S = \sqrt{{S_x}^2 + {S_y}^2}$	Nihai kenar görüntüsü (**S**), bu eşitlik kullanılarak hesap edilecektir.