Tuesday, 07-01-2014

SWE315 : C++

Homework 7 (there was NO 6) - (Due date 07-08 : a Week!!)

Solution:

1. Please send solution to: zbaharav@cogswell.edu
2. You know the drill by now: Simply hit reply, and no zipped directories etc.. Just ascii-files or Word documents (or equivalent)
3. Please attach as well (word document preferred) screen shots of the console output, and of the images produced.
4. This is **an important homework**, as we will build upon this one to extend to base-classes, inheritance, template, and so on. Please put the time to get this one right!

====

(see also hints-and-guidelines below!)

1. Create an Image class for dealing with ASCII pgm files, that will have the following variables and interfaces:
    a. Variables:
        i. String containing image name. E.g. “baboon_ascii.pgm”.
        ii. String containing File Format. E.g. “P2”.
        iii. String containing the comment string in the file (if any). E.g. “# Created on the 10 day of January.”.
        iv. Integers for numbers of rows, columns.
        v. Integer for the maximum gray level.
        vi. A two-dimensional integer array to hold the image.

    b. Interfaces:
        i. Constructor :
            1. Empty constructor
            2. Constructor from a file name. This will populate the fields, and most importantly fill in the array values.
        ii. Destructor, to clear memory allocated.
        iii. setter() and getter() function for all the variables (other than the 2D array). For example:
            1. `std::string getName();`
            2. `std::string setName(std::string str);`
            3. `std::string getComment();`
            4. `std::string setComment(std::string str);`
        iv. Two input/output functions:
            1. `bool ReadFromFile(std::string fileName);`
            2. `bool SaveToFile(std::string fileName);`
        v. One utility program :
        `void Pixelize(int s);`

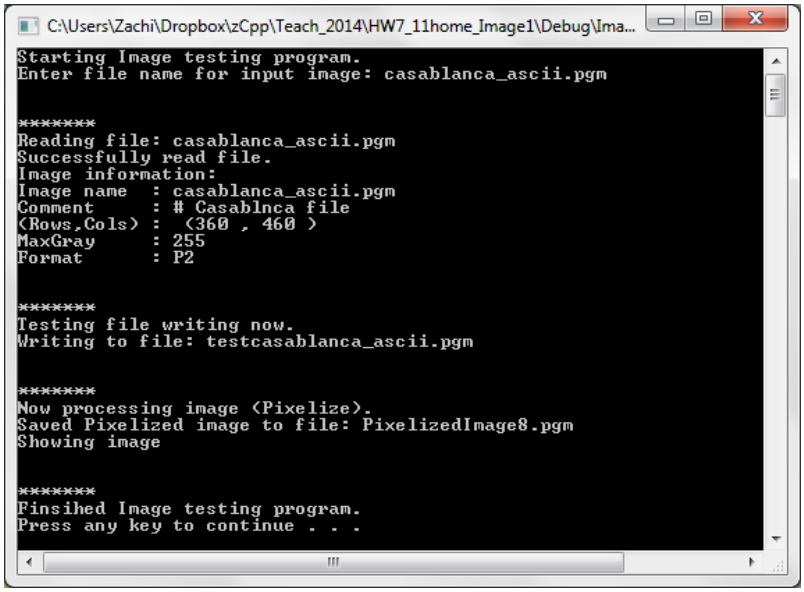
This utility program replaces ALL the pixels in an S-by-S blocks of the original with the same value. The value is the minimum value found in this S-by-S block of the original image. Here is an example of the first 3 rows (and the first part of those) of the Casablanca image, pixelized with s=3.

```
P2
# Casablnca file
460  360
255
146  146  146  142  142  142  137  …
146  146  146  142  142  142  137  …
146  146  146  142  142  142  137  …
:
```

And below you can see some examples of the results.

Make sure it all works on Casablanca_ascii.pgm and baboon_ascii.pgm images.

Screen shots of the console and images are given below.



Casablanca_ascii.pgm

Pixelized with factor of 5:
PixelizedImage5.pgm

Pixelized with factor of 8:
PixelizedImage8.pgm

Hints and guidelines:

1. You should know how to create the simple class with variables and methods.
2. You will need to dynamically allocate a 2D array.
3. Remember to clear this space in the destructor.

4. You already wrote a program to read and write a pgm file. Use the very same principles (aka cut-and-paste) in this case.
5. Attached you will find a few images to help.
6. The console output of your program should look like the above. Don't worry about word-for-word, but at least similar enough.

=== End of Homework 7 ====

```
// main : Testing program for class Image1
#include <iostream>
#include <string>

#include "Image1.h"

using std::cout;
using std::cin;
using std::endl;


int main(void)
{

	cout << "Starting Image testing program.\n";

	std::string fln;
	cout << "Enter file name for input image: ";
	cin >> fln;

	cout << "\n\n*******\n";
	cout<< "Reading file: "  << fln << endl;
	Image1 inputImage(fln);
	cout << "Successfully read file.\n" ;

	cout << "Image information:\n" ;
	cout << "Image name  : " << inputImage.getName() << endl;
	cout << "Comment     : " << inputImage.getComment() << endl;
	cout << "(Rows,Cols) :  (" << inputImage.getRows() << " , " <<
inputImage.getCols() << " )" << endl;
	cout << "MaxGray     : " << inputImage.getMaxGrayLevel() << endl;
	cout << "Format      : " << inputImage.getFileFormatString() << endl;


	cout << "\n\n*******\n";
	cout << "Testing file writing now.\n";
	// Write the file, and test your whole read/write seq.
	std::string test_fln = "test"+fln ;
	cout << "Writing to file: " << test_fln << endl;
	inputImage.SaveToFile(test_fln);

	// Does it show allright?
	system(test_fln.c_str());
```

```
	cout << "\n\n*******\n";
	cout<< "Now processing image (Pixelize).\n" ;
	// Modifying image
	int s=8;
	inputImage.Pixelize(int(s));
	std::string pixelized_str = "PixelizedImage" + std::to_string(s) + ".pgm" ;
	if (!inputImage.SaveToFile(pixelized_str))
	{
		cout << "Failed to open file for saving" << pixelized_str << "
;\nExiting...." ;
		exit(1);
	}
	cout << "Saved Pixelized image to file: " << pixelized_str << endl;
	// Show the image
	cout << "Showing image\n";
	system(pixelized_str.c_str());

	cout << "\n\n*******\n";
	cout << "Finsihed Image testing program.\n";

	system("pause");
	return 0;

}
```

// Image1.h

```
#ifndef _IMAGE1_H_
#define _IMAGE1_H_

#include <iostream>
#include <string>

class Image1{

private:
	int** I;
	std::string imageName;
	std::string fileFormatString;
	std::string commentString;
	int rows;
	int cols;
	int maxGrayLevel;

public:
	// Constructors
	Image1() { imageName = commentString = ""; rows=cols=0;};
	Image1(std::string fileName);
	//Image1(string name, int rows, int cols, int maxGrayLEvel, int** data,
string fileFormatString="P2", string comment="");

	// Destructors
	~Image1();

	// Setter and Getter
	std::string getName() { return imageName; }
```

```
        std::string setName(std::string str) { return imageName = str;} ;

        std::string getComment() { return commentString;} ;
        std::string setComment(std::string str) { return commentString = str;} ;

        int getRows() { return rows; }
        int getCols() { return cols; }
        int getMaxGrayLevel() { return maxGrayLevel; }

        std::string getFileFormatString() { return fileFormatString; }
        std::string setFileFormatString(std::string str) { return fileFormatString
= str; }


        void Pixelize(int s);


        // Input/output operations
        bool ReadFromFile(std::string fileName);
        bool SaveToFile(std::string fileName);

};

#endif
```

// Image1.cpp

```
#include "Image1.h"

#include <iostream>
#include <fstream>
#include <string>

using std::cout;
using std::ifstream;
using std::ofstream;
using std::getline;
using std::endl;

Image1::Image1(std::string fileName)
{
        if ( !ReadFromFile(fileName) )
        {
                cout << "Failed to open file " << fileName << "  ;\nExiting...." ;
                exit(1);
        }
}

Image1::~Image1()
{
        // Allocate dynamic memory

        // deAllocate room for each column
        for (int ii=0; ii<rows ; ++ii)
                delete [] I[ii];

        // deAllocate room for the rows
```

```
	delete [] I ;
}

bool Image1::ReadFromFile(std::string fileName)
{
	imageName = fileName;
	// Open files
	ifstream fin(fileName);
	if (!fin)  // fin.fail()
		return false;

	// Assume file header is like the following:
	/*
	P2
	# comment line
	512  512    <--- first number is cols, second is rows
	255   <-- range of White
	*/
	// Get the P2 line
	getline(fin, fileFormatString);

	// Get the comment line
	getline(fin, commentString);

	// Get the size of file
	fin >> cols >> rows;

	// Get the max gray level
	fin >> maxGrayLevel;

	// Allocate dynamic memory
	// Allocate room for the rows
	I = new int* [rows];

	// Allocate room for each column
	for (int ii=0; ii<rows ; ++ii)
		I[ii]=  new int[cols];


	// read the data
	for (int rr=0; rr<rows; ++rr)
		for (int cc=0; cc<cols; ++cc)
			fin >> I[rr][cc];

	fin.close();

	return true;
}


bool Image1::SaveToFile(std::string fileName)
{
	// Open files
	ofstream fout(fileName);
	if (!fout)  // fout.fail()
		return false;

	// Assume file header is like the following:
```

```
	/*
	P2
	# comment line
	512  512    <--- first number is cols, second is rows
	255   <-- range of White
	*/
	fout << fileFormatString << endl;

	// Set the comment line
	fout << commentString << endl;

	// Set the size of file
	fout << cols << "  " << rows << endl;

	// Set the max gray level
	fout << maxGrayLevel << endl;

	for (int rr=0; rr<rows; ++rr)
	{
		for (int cc=0; cc<cols; ++cc)
		{
			fout << I[rr][cc] << "  " ;
		}
		fout << endl;
	}

	fout.close();

	return true;
}


void Image1::Pixelize(int s)
{


	for (int rr=0; rr<(rows-s); rr+=s)
		for (int cc=0; cc<(cols-s); cc+=s)
		{
			int val = maxGrayLevel;
			for(int ii=0; ii<s ; ++ii)
				for(int jj=0; jj<s ; ++jj)
				{
					int tmp = I[rr+ii][cc+jj];
					val = (val < tmp) ? val : tmp;
				}
			for(int ii=0; ii<s ; ++ii)
				for(int jj=0; jj<s ; ++jj)
					I[rr+ii][cc+jj] = val;
		}

}

// END
```