Lab 03

ECE 1305, Ian Scott-Fleming

Nested Loops, file input and output, random numbers and trig functions

Sample Programs

As always, I'm providing you with some sample programs to help get you started on the assignment. Look them over, make sure you understand them, and refer to them as needed to get your own code working, but don't just cut/paste, or you'll not learn what you need to for this set of tasks.

(lab_03_sample_code.zip)

Here's a sample program that shows how to nest one loop inside another. The method is handy for all sorts of things, especially for producing tables, and outputting 2-D arrays (like images).

It produces a table like this:

```
0    1    2    3
10   11   12   13
20   21   22   23
30   31   32   33
40   41   42   43
50   51   52   53
60   61   62   63
70   71   72   73
80   81   82   83
90   91   92   93
```

You're probably heard of the "ASCII code" (American Standard Code for Information Exchange). The ASCII code is the standard computer code for representing text symbols for letters, numbers, punctuation, etc. with numbers between 0 and 127. Here's a sample program which uses a nested loop to output the ASCII table, in 16 rows of 8 columns. By the way, this program makes use of the function isprint( ), one of many useful character-related functions available in C++; it tells you whether the char is a printable char or not. (It needs the header file <cctype>). Have a look at the list of other char functions available, at

http://www.cplusplus.com/reference/cctype/ . Other useful functions there are things like isspace( ), isupper( ), toupper( ), etc., which tell you whether a char is a whitespace or an upper-case character, or convert one to uppercase.

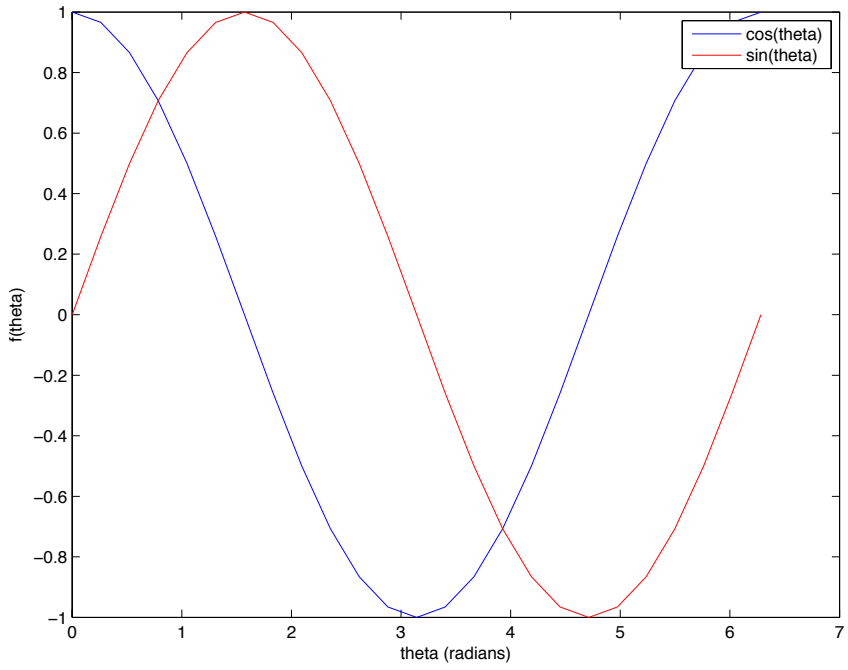
Now some sample programs demonstrating simple File I/O.

This one shows how to write data to a file. Study the program, compile and run it, then find the file in your file system and look at the output. Does it do what you expect it to?

This one shows how to read data from a file. It looks for a file called "numbers.txt", reads a series of numbers from the file, sums them and prints the average. *Note: it reads an arbitrary-sized file using a loop. Study it and read the comments and be sure you understand how it works.* Here's a numbers.txt file to test it with. (If you ran the previous sample program inside of Visual Studio, then finding the output file from the previous sample program will show you where Visual Studio writes its output files; put numbers.txt in the corresponding folder for this sample program, and it should find it to read from.)

This one shows how to write to a file, close it, and read from the same file.

Finally, since we're engineers, we should do something mathematical. C++ provides a rich set of trig and other math functions. So, here's a program that shows you how to get the sine and cosine of an angle. It prints a table of sines and cosines for angles from 0 to 2 PI radians (C++ trig functions work in radians, not degrees.) If you plot the output using excel or matlab, you'll get something like this



(Displaying graphics in a C++ program needs a Windows app (or a library like OpenGL's GLUT), but there are plenty of other tools we have available to us for graphing, like Excel and matlab. Here's a matlab script to read the file and plot it.)

Assignment

Part 1.. Writing data to an output file, reading data from an input file.

a. Write a program which prompts a user for the length of two sides of a right triangle, and writes the numbers and the length of the hypotenuse to "triangle.txt". Be sure to close the file before exiting the program.

b. Write a second program to open the same file ("triangle.txt"), read the three numbers back in, print them to the console, and then prints the area of the rectangle (1/2 the product of the first 2 sides). You'll need to make sure the text file from the first program is in the appropriate folder of the file-input project in order for your program to be able to open it properly.

Answer the following questions in the comments at the top of your program: If you run your output program more than once, and type the same input each time, do you know whether it worked properly the second time? Explain in your comments how to verify that it works properly every time.

Name your programs ***triangle_out.cpp and triangle_in.cpp***

Part 2. Nested Loops

a. Write a program, using nested loops, which outputs the numbers from 1 to 80 in 8 columns and 10 rows. Your output should look something like this. (The sample programs should help you figure out how to do this):

```
  1  2  3  4  5  6  7  8
  9 10 11 12 . . .    16
…
 73 74 75 76 77 78 79 80
```

b. Now modify it so it outputs the table like this. Note that the numbers go **down** the columns now (but you still have to output row by row…). You may need to do some thinking to figure out how to do this; it might help to do it on paper first, and figure out what you need to add as you go across the columns, and as you go down to the next row. (Hint: for each row, what is the value at the beginning of the row, and what do you need to add to each column to get the value for the next column of the current row?)

```
1   11   21   31   41   51   61   71
2   12   22   32   42   52   62   72
```

```
…
 10   20   30   40   50   60   70   80
```

c. Now modify it so it prompts you for the maximum number to output, and the number of rows. (Keep it simple…no need for error checking on the input.)

```
How many numbers to output?  80
how many rows? 14
  1   15   29   43   57   71
  2   16   30   44   58   72
  3   17   31   45   59   73
  4   18   32   46   60   74
  5   19   33   47   61   75
  6   20   34   48   62   76
  7   21   35   49   63   77
  8   22   36   50   64   78
  9   23   37   51   65   79
 10   24   38   52   66   80
 11   25   39   53   67
 12   26   40   54   68
 13   27   41   55   69
 14   28   42   56   70
```

Test it first with a # of rows that divides evenly into the # of numbers (e.g. 5 columns, 50 numbers). Then test it with numbers that don't fit evenly (e.g. 14 rows, 80 numbers. Did your program work properly?

Integer arithmetic and the modulus operator come in handy here for figuring out how many columns to output and how much to add when moving to the next column. Note that if the number of rows doesn't divide evenly into the total number to output (remember lab 2?), you'll have to do something creative to avoid writing more numbers than you should. This will require a little thought, and the use of an if/then/else.

You only need to turn in part C for grading. However, **do the program incrementally, as I have assigned, so you only have a manageable amount to stumble over and get working before you move on.**
Also: don't delete or overwrite your parts a or b programs, as you'll want to be able to refer back to them (or start over from them) if you run into problems on the next step.

*Name your program **columns.cpp***

Part 3: Digital Images

You will not only learn to program C++ in this class, we will also show you some other things. One of these things is the plain PGM image file format. You will be working with this image format in this class. First, read this short introduction (doc version) to digital image and the PGM image format.

Many applications have PGM support built-in. Unfortunately, neither Windows nor Mac OS X currently come with an app which can display PGM images. On Windows, you can download IrfanView if you need a viewer. It is highly recommended that you have a program to view these files. For Macs, you can use ToyViewer.

For Irfanview, uncheck "Use Resample for fitting" and "Use Resample for zooming" under "View | Display Options". Otherwise (by default) Irfanview blurs images when you zoom them. That's great for general image viewing, but awkward for programmers trying to figure out what their program just wrote out to a PGM file.

The complete specification for the PGM format can be found on the netpbm website (look for plain PGM) or in Wikipedia: Portable Pixmap. A plain PGM file is just a simple text file that follows the following standard:

- The first line is P2
- The second line specifies the size of the image (width and height, in number of pixels)
- The third line specifies the maximum gray value (maxval) This is the largest number that can appear in the actual image data
- After that, each line consists of a line of image data, given in gray-values where 0 is black and whatever the maxval is is white. [1]
- (These are grayscale images; there is a "P5" color version called PPM; see the netpgm site above for more information.)

given all this, here's an example:

```
P2
```

[1] the pixel data is actually free-format – it doesn't matter how many pixel values you put on a line, just so long as there are as many pixel values as there are pixels in the image (ncols * nrows)

Technically, lines should be no longer than 80 characters, but every program I've encountered that can read PGM images is happy if there are more than 80 chars per line, so it is often easiest just to put an entire row of pixels on the same line in the file.

```
14 7
15
0  0  0  0  0  0  0  0   0  0  0  0  0  0
0 10  0  0  0 10  0  0  15 15 15  0  0  0
0 10 10  0 10 10  0  0  15  0  0 15  0  0
0 10  0 10  0 10  0  0  15 15 15  0  0  0
0 10  0  0  0 10  0  0  15  0  0 15  0  0
0 10  0  0  0 10  4  0  15 15 15  0  4  0
0  0  0  0  0  0  0  0   0  0  0  0  0  0
```

Or you can download: mb.pgm

You can also read and display PGM images in matlab with a simple script like this:

```
I = imread('mb.pgm');
imshow(I);
```

As an image, mb.pgm looks like this (if you zoom it about 40x). (It's Max Berger's initials...)

Part 3 assigment:

a. Using a text editor (not a C++ program) create a similar PGM file with your initials. Use the plain PGM file format as described here, but use 255 for the

maxval. Use a light gray (say, 180) for the background, and make each letter a different dark gray (128, 64, 32) or black (0).

For creating and editing the PGM, you can use notepad (on Windows), textedit (on OS X, but be sure to save as plain ascii, not the default of rtf, rich text format), or using Visual Studio. Be sure to give your file the extension of ".pgm" . *Note: if you aren't displaying all file extensions in Windows' file browser, your file will probably be called "initials.pgm.txt", which won't be recognized as a PGM image because it has the wrong extension.*

b. Write a C++ program that writes a PGM out to a text file. Make your image 30 columns by 20 rows. Set the maxval to 255. Draw a triangle on a dark gray background, making the triangle brighter towards the bottom:

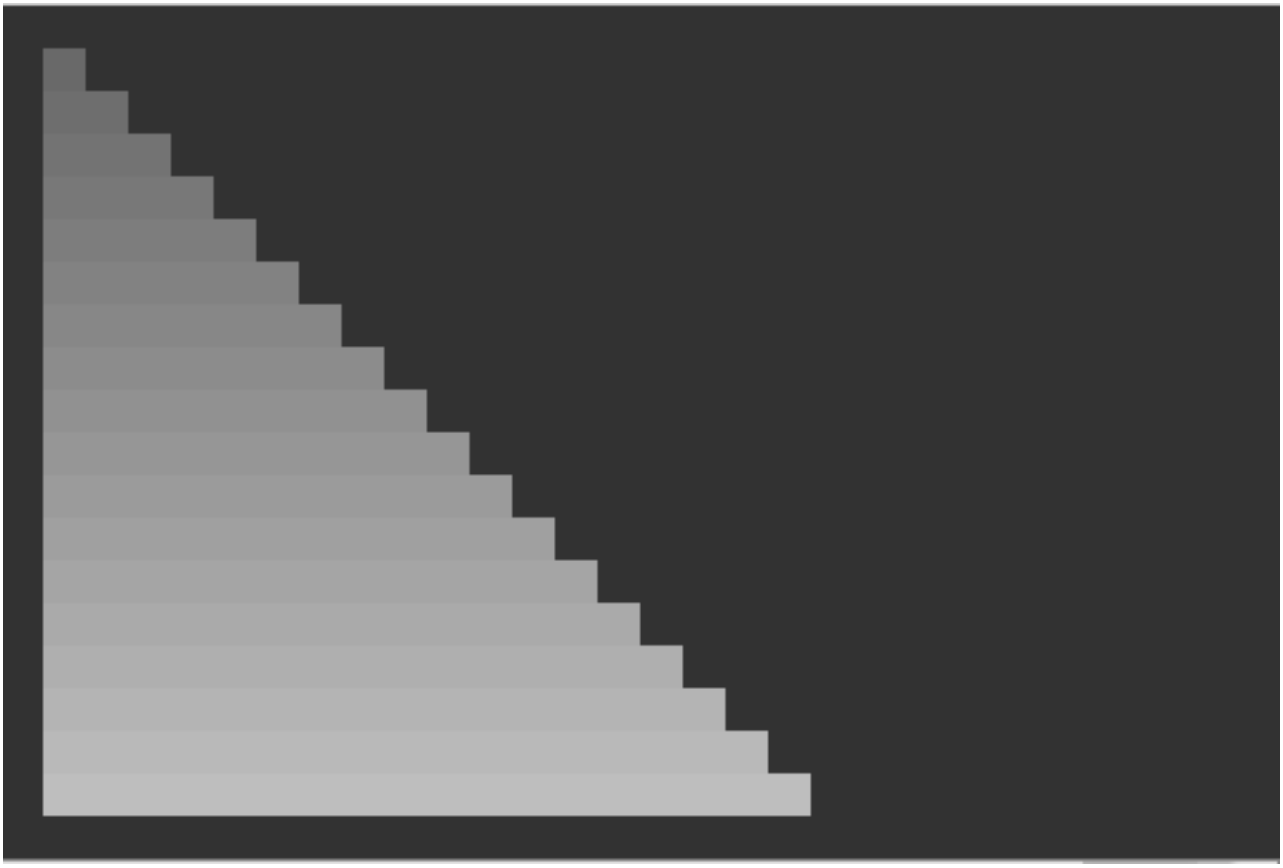
```
P2 30 20 255
 50  50  50  50 ...
 50 100  50  50 ...
 50 105 105  50 ...
 50 110 110 110 ...
. . .
 50  50  50  50 ...
```

This is really simpler than it may look. The steps are:

1. prompt for filename, read it in, and open the file
2. Write out the header, with something along the lines of
   `<< "P2" << " " << ncols << " " << nrows << " " << maxval << endl;`
3. Use a set of nested loops:
   note: if you count from 0 to nrows-1, then the counter has the # of values to output
   a. for each row:
      i. if drawing the 1st or last rows, output a row of all background values. Otherwise
      ii. output the columns of the row:
         1. output a 0
         2. output the desired # of non-zero pixel values
         3. output enough background pixels to complete the row
      iii. write out an endline (endl)

Make sure to output the correct number of pixels in each row. If the row is to long or too short, your triangle won't be a right-triangle.

Viewing your PGM in a viewer (or with matlab), it should look something like this (zoomed about 16 times) :



Name your program ***make_pgm.cpp***

Challenge:

The challenge is optional, but is a fun program to write, so tackle it if you have time and the inclination.

Write a program that plots a circle in the middle of a black PGM. For this, it may be helpful to use the sqrt( ) function, which returns the square root of a number. For example:

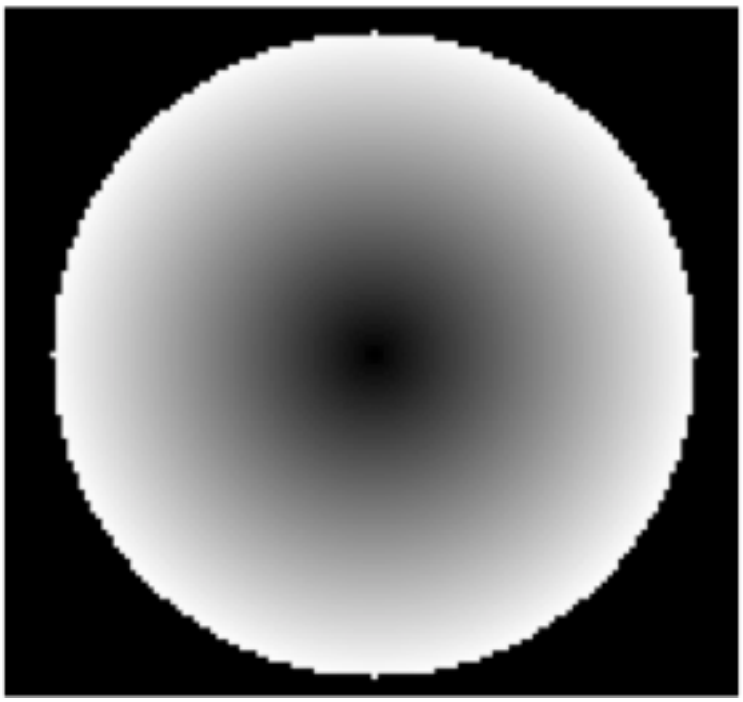
```
y = sqrt(2.0);
```

sets y to ~1.414. If the distance of a pixel from the center of the circle is less than the radius, you're inside the circle. Otherwise you're outside.

For more excitement (this IS exciting, isn't it?) you could shade the the circle as you get closer to the center. If that's still not exciting enough, try prompting the user for

a box size and draw a box in the the middle of the circle. Here's a circle example, with shading towards the center of the circle:

```
pgm name? circle.cpp
ncol, nrow? 128 120
radius? 56
done writing PGM to circle.pgm
```



Extra credit for all who try, and a prize for the first good solution submitted (code must be nicely formatted and commented). Email me if/when you posted your circle program.

Note: for project 1, you'll calculate a ball's position over time (lab 02), and for project 2, you'll draw it in a PGM, and for project 3 we'll animate it and see if you can create a video game. So, figuring out now how to draw a circle in a PGM will be a good start.

If you submit a solution, call your program ***circle.cpp***

What to submit:

You should submit 4 programs and your intials.pgm file:

1. triangle_in.cpp

2. triangle_out.cpp
3. columns.cpp
4. initials.pgm
5. make_pgm.cpp

and if you try the challenge, also submit

6. circle.cpp

Dept. of Electrical and Computer Engineering
Texas Tech University