Lab 2 – Pointers and File I/O
CPSC 1021- Fall 2017

CLEMSON
School of COMPUTING

Introduction

In today's lab, we will be calculating the center of mass of a set of objects that you will read in from provided data files. You will then output the location of the center of mass to a file, which you will be able to plot using the **gnuplot** plotting tool.

Lab Objectives

- Manipulate C style pointers
- Read in and Write out to files
- Use standard I/O functions (scanf, printf, etc)
- Use the math.h library and gcc linker commands
- Use command line arguments

Prior to Lab

If you are using a VirtualBox VM, install **gnuplot** using the command:

sudo apt install gnuplot

If you are using a lab machine, this software is already installed.

Lab Instructions

You will be reading in **gnuplot** data files that describe circles. The names of input and output files **will be given as command line arguments**. They look like this:

```
#7
1.3  1.4  1.0
2.1  2.9  2.5
3.5  2.6  0.5
4.3  1.7  1.0
2.8  2.3  2.0
4.3  1.3  1.5
1.9  1.8  0.5
```

The first line is a comment which denotes how many circles are described in the file, where each subsequent line consists of an x coordinate, y coordinate, and radius. For this lab, you will be provided the following files:

lab2.c
plot_circles.plg
plot_centroid.plg
circles.dat

You will be implementing **3** functions: main, read_data, and calc_centroid.
The skeleton for these functions and descriptions of their purpose can be found in **lab2.c**

Pointer Review

A pointer is a type of variable that allows you to specify the address of a variable. A pointer provides a convenient means of passing arguments to functions and for referring to more complex datatypes such as structures. Pointers are also essential if you want to dynamically allocate memory.

The specific value in a pointer is seldom important, as long as it contains the address of the variable that you want. You need to declare and initialize pointers just as you would other variables, but there are special operators that you need to use.

The following table shows the special characters used in C to declare and use pointer

*	Dereference/indirection operator	This operator is used to declare a variable as a pointer. It is also used to access the value pointed to by the pointer variable.
&	Reference or address-of operator	This operator is used before a variable to indicate the address of that variable.

Simple Pointer Use

- Once a pointer is declared, the programmer must initialize the pointer, that is, make the pointer point to something. Bad things can happen if you don't ensure that all pointers are initialized when declared. At a minimum initialize the pointer to NULL or 0.
- Like regular variables, uninitialized pointers will not cause a compiler error, but using an uninitialized pointer could result in unpredictable and potentially disastrous outcomes.
- Until a pointer holds an address of a variable, it is **useless**.

Remember that you can treat a pointer exactly like an array, and using the **malloc** command allows you to request a dynamic amount of memory to hold data. Be aware of how to use such a structure and to access its elements.

File I/O

For this lab, we will use standard C input/output functions. This includes fopen, fprintf, fscanf, etc.

Please review these functions and how to open files:

http://www.cprogramming.com/tutorial/cfileio.html

The Centroid or Center of Mass

The following is sourced from: http://hyperphysics.phy-astr.gsu.edu/hbase/cm.html

For a given set of masses **m**, the combined center of mass is given component-wise with the following formulas:

$$x_{cm} = \frac{\sum_{i=1}^{N} m_i x_i}{M} \qquad y_{cm} = \frac{\sum_{i=1}^{N} m_i y_i}{M} \qquad z_{cm} = \frac{\sum_{i=1}^{N} m_i z_i}{M}$$

Where **M** is the total mass of every object. For our purposes, we are going to treat the **AREA** of each circle described in the data files as the mass of the object (note that for objects of uniform density, mass would be directly correlated with area anyway).

Your program should:

1. Read in the input data file and store the circles in a dynamically sized array (using pointers)
2. Calculate the center of mass in the X/Y plane. Use the x and y locations of the center points and use the **area** of each circle as its mass (**little m)** value
3. Output the location of the centroid in a new file (given as a command line argument)

The output file will consist of two whitespace separated numbers, i.e.

centroid.dat:
2.345 2.3984

Math.h

To get the value of Pi or use the power function in C, make sure to include the math.h header.

Having included this header, you could calculate area of a circle as follow:

double area = M_PI * pow(r,2);

GNUPLOT

We have provided two **gnuplot** scripts that will allow you to visualize the circles/centroid once you have generated the output data.

Simply run the command:

gnuplot plot_circles.plg *or* **gnuplot plot_centroid.plg**

These scripts expect the circle data to be store in **circles.dat**, but you can specify a different circle file with the following command:

gnuplot -e "circlefile='filename'" plot_circles.plg

Compile and Execute

Use GCC to compile your code as follows:

gcc lab2.c -Wall -lm -o centroid

Execute the program

./centroid <input_file> <output_file>

FORMATTING:

1. Your program should be well documented
2. Each file should include a header:
3. Your program should consist of proper and consistent indention
4. No lines of code should be more than 80 characters

```
// Sample Header
/******************
  your name
  username
  Lab 1
  Lab Section:
  Name of TA
******************/
```

5 – 10 points will be deducted for each of the above formatting infractions.

Submission Instructions

- Test your program on the School of Computing server prior to submitting.
- Use the tar utility to tar.gz all source files. **Do not tar an entire directory! When I untar your archive, I should see all of the files you included, not a top level directory! Failure to correctly tar may result in up to a 25 point penalty!**

  EX. tar –czvf nglyder-lab2.tar.gz lab2.c

- Name your tarred file **<username>-lab<#>.tar.gz** (ex. nglyder-lab2.tar.gz)
- Use handin (http://handin.cs.clemson.edu) to submit your archive