# Lab 4

## Reading and Writing Files

Skills Required

* Read and write Files
* Work with arrays and vectors, Create Functions, Include Headers and other files, Loops (while, for), Conditional (if, switch), Datatypes, assignment, etc.

1. Assignment

You may have learned the distance formula in math or physics class. It lets you know the distance between any two points in space using x- and y-coordinates in a two-dimensional space. The same formula can be modified to work in three-dimensional space if you know both sets of x, y, and z coordinates:

This lab will read in several sets of point coordinates from a file calculate their distances, then write those calculated distances to a file.

**Introduction to Reading and Writing to Files**

Using ifstream (input file streams) and ofstream (output file streams) variables, you can load in data from one file and out to another. First, you need to create two variables called fin and fout which will correspond to your input and output files:

ifstream fin("input.txt");

ofstream fout("output.txt");

You can stream in data to or from your variables from input or output files in the same way you use cin and cout to get data to and from the console (keyboard/screen):

fin >> x; // Reads in data from input.txt to the variable x

fout << x; // Writes out data from the variable x to output.txt

The first line reads in one number from the input file (input.txt) and saves it in the variable called x. The second line writes the value of x to the output file (output.txt). You can use these ideas to read in values and store them. You can stream in or out several variables at the same time:

fin >> x >> y >> z;  
 fout << x << y << z;

When reading in data, you often use loops to continue to read in data until you meet certain conditions. Some examples include:

// If you know how many data elements you are reading in  
 for (int i = 0; i < 5; i++) { // Ex: reading in 5 things  
 fin >> x;

}

// If you don’t know how many data elements you are reading in

while (fin.good()) { // While the file still has elements to read in

fin >> x;

};

**Pitfall:** The istream’s .good() method returns true if the last attempt to read something was successful. This means it’s possible to read the last item from a file and not recognize you’re at the end of the file. It’s not until another iteration of the above loop happens, and the fin>>x fails, that end of file is detected; and then, of course, you’re already in the loop body. The symptom of this is that the program appears to read the last element of the file twice. There are several ways to address this. One is to restructure the loop slightly, using a priming read:

fin >> x;   
while (fin.good()) {   
 // process value of x we just read   
 fin >> x; // try reading item for next iteration  
}

Another is to use the return value from the read statement itself:

while (fin >> x ) {   
 // process value of x   
}

Or add an explicit test:

while (fin.good()) {   
 fin >> x;   
 if (fin.good()) {   
 // process x   
 }  
}

**Writing the Distance Function**

You need to write a function called calcDistance(). It should take in 6 parameters (all doubles which correspond to the two x, two y, and two z coordinates), calculate the distance between the two points, and return that calculation to the main() function as a double.

Obviously there’s no way to type in a into your program, but C++ has a built-in function called sqrt() that looks like this:

double sqrt (double x);

It takes in a double >= 0 and returns the square root back.

There is a function to do exponents as well. You can use the pow() function like so:

double pow (double x, double y); // Corresponds to xy

You may need to #include the <cmath> library to have access to these functions. There is another way to square numbers. Just multiply the number twice (like x \* x).

Breaking down the formula, you should be able to figure out how to take several values and build a function out of it.

**Storing in Vectors**

Store your input data in vectors of points; these will be either a vector of vectors of integers; or you can define a struct called Point and make a vector of Points. Remember you can resize a vector or use push\_back to add a new value to a vector. This keeps your program easy to maintain if the number of values in the input file changes. The input file that the grader uses may have a different number of input lines to work with. Your output file should be the distance between each pair of points, each on a line by itself.

**Sorting the Values (Stretch Goal)**

Write out to a new file called output\_sorted.txt all the distances sorted from lowest to highest. You can use any of the sorting algorithms found in your text. For this you’ll want to write a sorting function that takes a vector passed by reference.

**Submit your assignment**

Save all your files, commit and push everything to GitHub, and submit the link on Canvas.