ECE 3220 Lab6
Introduction to C++ and Classes

Marshall Lindsay
mblgh6@mail.missouri.edu
14211662

3/14/2017

Objective:

The objective for this lab was to become familiar with classes, objects, constructors, overloading and exception handling.

Discussion:

This lab was essentially converting lab4 to a C++ program. Each function to manipulate data was converted into a method of the Signal class. This class held information about the signal whose data points are found in the data files. After the data was gathered and manipulated by the methods per the user's instruction, the updated data could be saved to a file. Every input provided by the user is checked for validity.

Experiments:

There was a large testing phase with this program that included inputting improper inputs, correct inputs and checking for proper outputs. Figure1 shows every stage of user input that is put through proper error checking. Figure2 shows the content of default_data.txt, the data file used for the output screenshots. Figures 3 - 6 show the data saved after scaling, offsetting, centering, and normalizing respectively.

Conclusion:

By the end of this lab a concrete understanding of classes, methods, overloading and exception handling was obtained.

Figures:

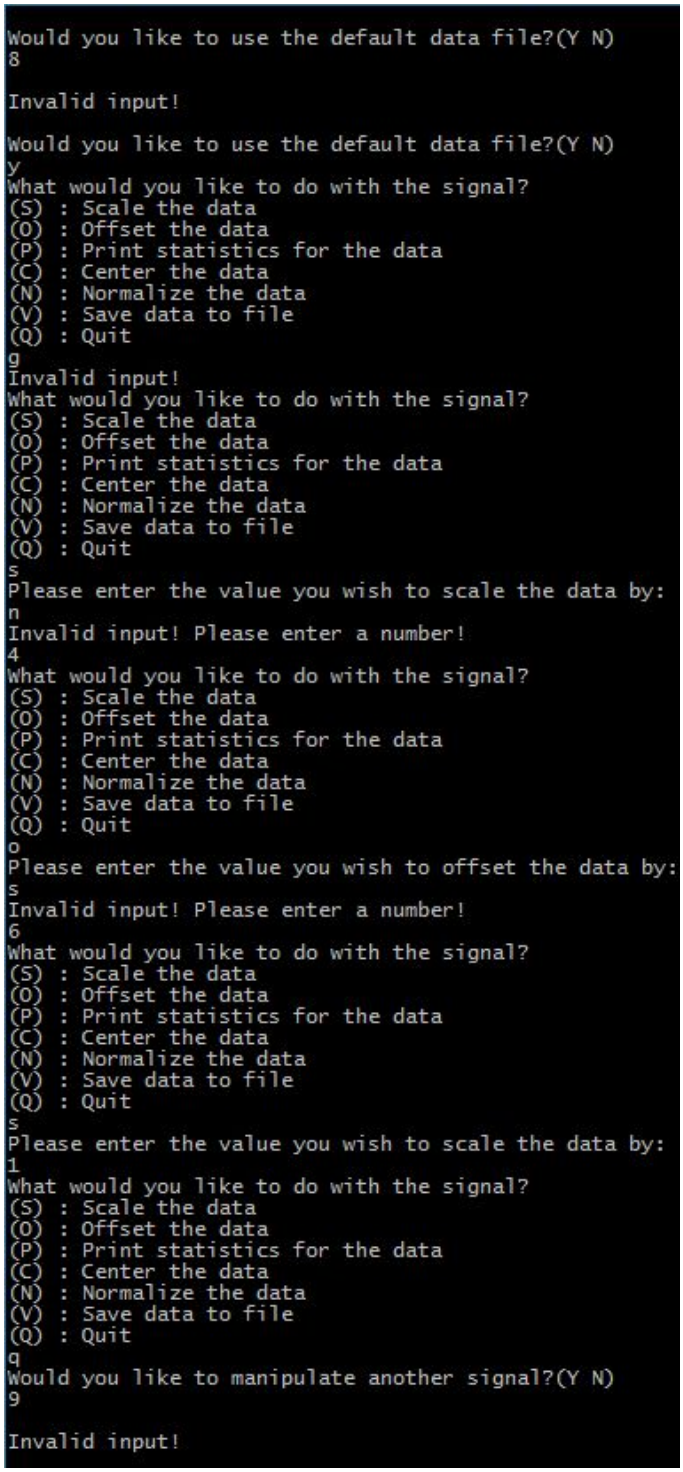

Figure 1: Error checking

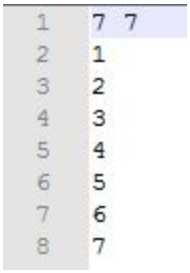

Figure2: Default_Data.txt

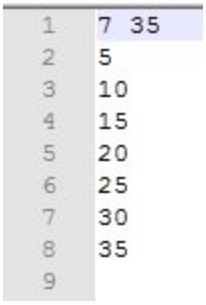

Figure3: Scaled output of Default_Data.txt with the value 5

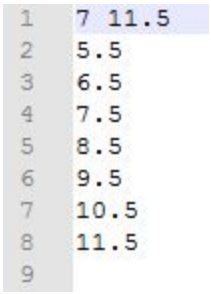

Figure4: Offset output of Defualt_Data.txt with 4.5

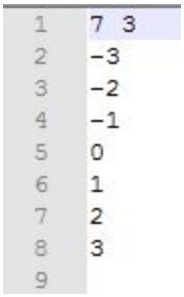

Figure5: Centered output of Default_Data.txt

```
7 1
0.142857
0.285714
0.428571
0.571429
0.714286
0.857143
1

```

Figure6: Normalized output of Default_Data.txt

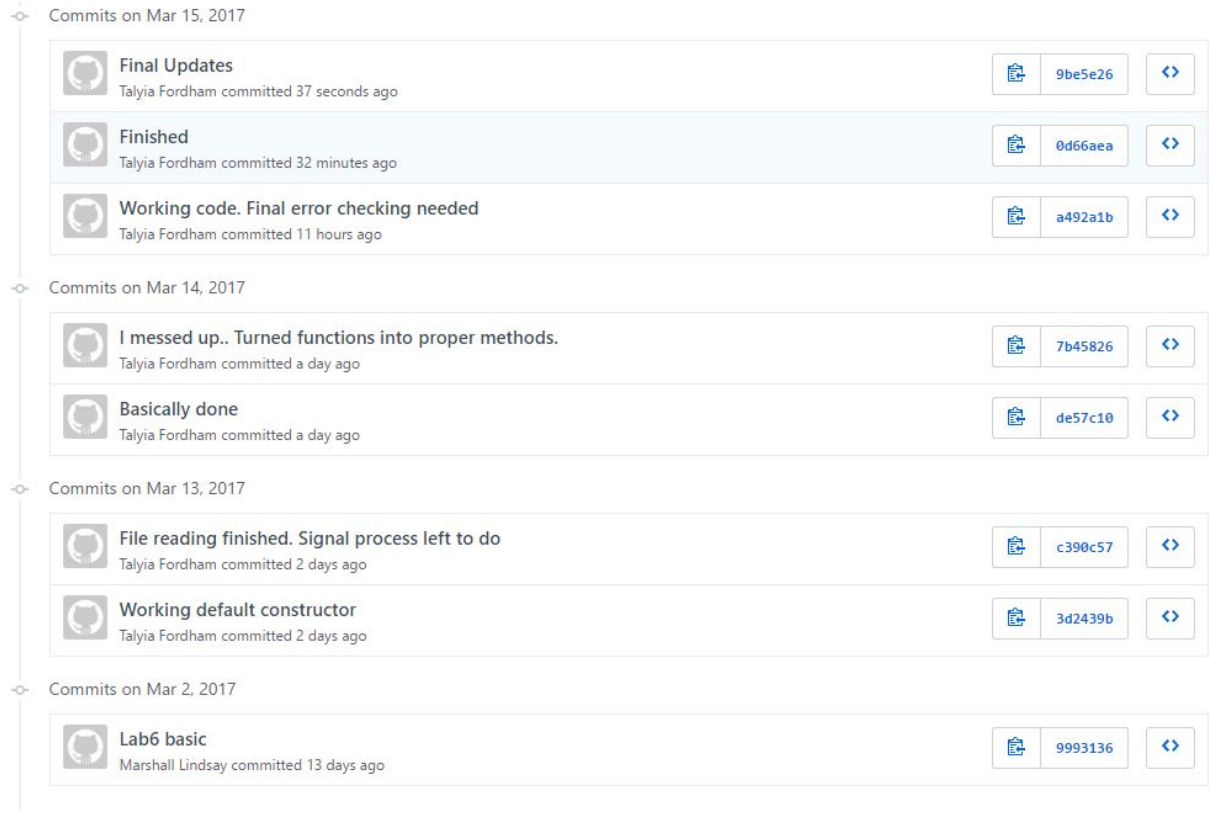

Figure7: GitHub Commits

Source Code:

```
/*Marshal Lindsay
 *14211662
 *ECE32220
 *Lab6, Introduction to C++ and Classes
*/


#include <iostream>
#include <fstream>
#include <string>
using namespace std;
```

```
#define DEFAULT_DATA_FILE "defaultData.txt"
//Signal Class
class Signal{
	private:
		string fileName;
		int length;
		double max_value;
		double average;
		double* data;

	public:
		void calcAvg();
		void scale(double);
		void offset(double);
		void center();
		void normalize();
		void Sig_info();
		void Save_file(string);
		Signal();
		Signal(int);
		Signal(char*);
		~Signal();

};
//Destructor
Signal::~Signal(){
	delete [] data;

}
//Default Constructor
Signal::Signal(){

	fstream dataFile;
	int i;
	this->fileName = DEFAULT_DATA_FILE;
	//Open the default data file
	dataFile.open(DEFAULT_DATA_FILE);

	//Check that the file was opened correctly.
	//If not, set default values and return
	if(!dataFile.is_open()){
		cout<<"\nCould not open "<<DEFAULT_DATA_FILE<<" setting default values!"<<endl;
		length = 0;
		max_value = 0;
		average = 0;
		data = NULL;
		return;
	}

	//Read the number of integers in the datafile
	dataFile >> length;

	//Allocate memory for the data array.
```

```
	data = new double[length];

	//Check that data was allocated correctly.
	//If it wasn't, set default values and quit.
	if(data == NULL){
		cout<<"Could not allocate memory for data in default constructor!"
			<<"setting default values and exiting!"<<endl;
		length = 0;
		max_value = 0;
		average = 0;
		data = NULL;
		return;
	}

	//Read the max value from the data file
	dataFile >> max_value;

	//Read the data from the file and place in the data array.
	for(i = 0; i < length; i++){
		dataFile >> data[i];
	}

	//Close the data file.
	dataFile.close();
	calcAvg();
	return;
}
//Integer Input Constructor
Signal::Signal(int L){
	fstream dataFile;
	int i;
	char ones, tens;
	string fileName;

	//Declare a const char* to be used with the .open() on line 115
	//that points to the file name.
	const char* ptrFileName = fileName.c_str();


	//Convert the integer L to 2 chacters to be appended to a string
	if(L < 10){
		ones = L + 48;
		tens = 48;
	}else if(L > 10 && L < 100){
		ones = L % 10 + 48;
		tens = L / 10 + 48;
	}else{
		cout<<"\nInvalid file number! Setting default values and quitting!"<<endl;
		length = 0;
		max_value = 0;
		average = 0;
		data = NULL;
		return;
	}
```

```
	//Append the converted integers into the file name template
	fileName = std::string("Raw_data_") + tens + ones + ".txt";

	this->fileName = fileName;
	//Open the correct file
	dataFile.open(ptrFileName);

	//Check that the file was opened correctly.
	//If not, set default values and return
	if(!dataFile.is_open()){
		cout<<"\nCould not open "<<fileName<<" setting default values!"<<endl;
		length = 0;
		max_value = 0;
		average = 0;
		data = NULL;
		return;
	}

	//Read the number of integers in the datafile
	dataFile >> length;

	//Allocate memory for the data array.
	data = new double[length];

	//Check that data was allocated correctly.
	//If it wasn't, set default values and quit.
	if(data == NULL){
		cout<<"Could not allocate memory for data in default constructor!"
			<<"setting default values and exiting!"<<endl;
		length = 0;
		max_value = 0;
		average = 0;
		data = NULL;
		return;
	}

	//Read the max value from the data file
	dataFile >> max_value;

	//Read the data from the file and place in the data array.
	for(i = 0; i < length; i++){
		dataFile >> data[i];
	}

	dataFile.close();
	calcAvg();
	return;
}
//Char* constructor
Signal::Signal(char* fileName){
	fstream dataFile;
	int i;
	this->fileName = fileName;
```

```
	//Open the data file
	dataFile.open(fileName);

	//Check that the file was opened correctly.
	//If not, set default values and return
	if(!dataFile.is_open()){
		cout<<"\nCould not open "<<fileName<<" setting default values!"<<endl;
		length = 0;
		max_value = 0;
		average = 0;
		data = NULL;
		return;
	}

	//Read the number of integers in the datafile
	dataFile >> length;

	//Allocate memory for the data array.
	data = new double[length];

	//Check that data was allocated correctly.
	//If it wasn't, set default values and quit.
	if(data == NULL){
		cout<<"Could not allocate memory for data in default constructor!"
			<<"setting default values and exiting!"<<endl;
		length = 0;
		max_value = 0;
		average = 0;
		data = NULL;
		return;
	}

	//Read the max value from the data file
	dataFile >> max_value;

	//Read the data from the file and place in the data array.
	for(i = 0; i < length; i++){
		dataFile >> data[i];
	}

	//Close the data file.
	dataFile.close();

	calcAvg();
	return;

}

void Signal::calcAvg(){
	double total = 0;
	int i;
	for(i = 0; i < this->length; i++){
		total += data[i];
	}
```

```
	this->average = (double)(total) / this->length;
	return;
}

void Signal::scale(double scaledValue){
	int i;
	this->max_value *= scaledValue;

	for(i = 0; i < this->length; i++){
		this->data[i] *= scaledValue;
	}
	calcAvg();

	return;
}

void Signal::offset(double offsetValue){
	int i;
	this->max_value += offsetValue;

	for(i = 0; i < this->length; i++){
		this->data[i] += offsetValue;
	}
	calcAvg();

	return;
}

void Signal::center(){
	int i;
	this->max_value -= this->average;
	for(i = 0; i < this->length; i++){
		this->data[i] -= this->average;
	}
	calcAvg();
	return;
}

void Signal::normalize(){
	int i;

	for(i = 0; i < this->length; i++){
		this->data[i] /= this->max_value;
	}
	this->max_value /= this->max_value;
	calcAvg();
	return;
}

void Signal::Sig_info(){
	//Print the information about the signal/data
	cout<<"\nNumber of data points (length): "<<this->length
		<<"\nMaximum value (max_value): "<<this->max_value
		<<"\nAverage of data (average): "<<this->average<<"\n"<<endl;
```

```
	return;

}
void Signal::Save_file(string newFileName){
	newFileName = std::string(newFileName) + ".txt";
	const char* newFilePtr = newFileName.c_str();

	//open save file

	ofstream saveFile(newFilePtr);
	int i;

	//Save the Length, Max Value, and the data to the save file
	saveFile << this->length<< " " << this->max_value<<endl;

	for(i = 0; i < this->length; i++){
		saveFile<< this->data[i]<<endl;
	}

	//Close the file
	saveFile.close();
	return;

}

/*FUNCTIONS*/
void optionMenu();
void helpMenu();
double scaling();
double offsetting();
string fileSave();



int main(int argc, char** argv){
	Signal* dataSignal;
	double value;
	int userInput1;
	char userInput2;
	char fileName[25];
	string newFileName;
	const char* newFilePtr = newFileName.c_str();


	cout<<"\n****Lab6****"<<endl;
	while(1){

		cout<<"\nWould you like to use the default data file?(Y N)"<<endl;
		cin >> userInput2;
		while( userInput2 != 'y' &&
			userInput2 != 'Y' &&
			userInput2 != 'n' &&
			userInput2 != 'N')
```

```
		{
			cout<<"\nInvalid input!"<<endl;
			cout<<"\nWould you like to use the default data file?(Y N)"<<endl;
			cin >> userInput2;

		}

	if(userInput2 == 'Y' || userInput2 == 'y'){
		dataSignal = new Signal;
	}else{
		cout<<"\nEnter 'f' to use the file name or 'n' to use the file number:"<<endl;
		cin >> userInput2;

		while( userInput2 != 'n' &&
		userInput2 != 'N' &&
		userInput2 != 'f' &&
		userInput2 != 'F')
		{
			cout<<"\nInvalid input!"<<endl;
			cout<<"\nEnter 'f' to use the file name or 'n' to use the file number:"<<endl;
			cin >> userInput2;

		}

		if(userInput2 == 'f'|| userInput2 == 'F'){
			cout<<"Please enter the file name to be opened:"<<endl;
			cin >> fileName;
			dataSignal = new Signal(fileName);
		}else{
			cout<<"\nPlease enter the file number to be opened:"<<endl;
			cin >> userInput1;
			while(cin.fail()){
				cin.clear();
				fflush(stdin);
				cout<<"Invalid input! Please enter a number!"<<endl;
				cin >> userInput1;
			}
			dataSignal = new Signal(userInput1);
		}
	}


//Data manipulation here

//Set a known value for the userInput2
userInput2 = 'p';
//Data manipultion loop
while(1){
	//Print the option Menu
	optionMenu();
	//Grab user input
	cin>> userInput2;
	//Convert input to uppercase
	if(islower(userInput2)){
```

```
                userInput2 = toupper(userInput2);
        }
        //Switch input and call appropriate functions

        switch(userInput2){
        case 'S':
                value = scaling();
                dataSignal->scale(value);
                break;
        case 'O':
                value = offsetting();
                dataSignal->offset(value);
                break;
        case 'P':
                dataSignal->Sig_info();
                break;
        case 'C':
                dataSignal->center();
                break;
        case 'N':
                dataSignal->normalize();
                break;
        case 'V':
                newFileName = fileSave();
                dataSignal->Save_file(newFilePtr);
                break;
        case 'Q':
                break;
        default:
                cout<<"Invalid input!"<<endl;
                break;
        }

        if(userInput2 == 'Q'){
                break;
        }

    }
    cout<<"Would you like to manipulate another signal?(Y N)"<<endl;
    cin >> userInput2;
    while( userInput2 != 'y' && userInput2 != 'Y' && userInput2 != 'n' && userInput2 != 'N'){
                cout<<"\nInvalid input!"<<endl;
                cout<<"Would you like to manipulate another signal?(Y N)"<<endl;
                cin >> userInput2;
        }
    if(userInput2 == 'n' || userInput2 == 'N'){
        delete dataSignal;
        break;
        }



    }
return(0);
```

```
}

void optionMenu(){

	cout<<"What would you like to do with the signal?"<<endl;
	cout<<"(S) : Scale the data\n"
		<<"(O) : Offset the data\n"
		<<"(P) : Print statistics for the data\n"
		<<"(C) : Center the data\n"
		<<"(N) : Normalize the data\n"
		<<"(V) : Save data to file\n"
		<<"(Q) : Quit"<<endl;
	return;
}

double scaling(){
	double value;

	cout<<"Please enter the value you wish to scale the data by:"<<endl;
	cin >> value;

	//Check that the user input is a double type
	while(cin.fail()){
		cin.clear();
		fflush(stdin);
		cout<<"Invalid input! Please enter a number!"<<endl;
		cin >> value;
	}

	return(value);

}

double offsetting(){
	double value;

	cout<<"Please enter the value you wish to offset the data by:"<<endl;
	cin >> value;

	//Check that the user input is a double type
	while(cin.fail()){
		cin.clear();
		fflush(stdin);
		cout<<"Invalid input! Please enter a number!"<<endl;
		cin >> value;
	}

	return(value);

}

string fileSave(){
	string newFileName;
```

```
        cout<<"Please enter the name of the file to save the updated data to:"<<endl;
        cin>> newFileName;

        return(newFileName);
}
```