```
//Thomas Gibbons
//Oct 9, 2016
#include <iostream>
#include <sstream>
#include <cstring>
#include <string>
#include <vector>
#include <stdio.h>      //sprintf
#include <stdlib.h>     //atoi
using namespace std;

class Signal{
    //friend Signal operator+(Signal,Signal);

    public:
        vector<int> signalData; //original data
        int Max;
        double average;
        int Length;

        vector<double> alteredData; //altered data
        double alteredMax;
        double alteredAverage;

        void dataReset();       //methods
        void maximum();
        void mean();
        int readFile(string);

        void operator+(double); //operators
        void operator*(double);
        double offsetVal;
        double scaleVal;

        void centerFile(){*this + (alteredAverage*(-1));}
        void normalizeFile(){*this * (1.0/alteredMax);}
        void saveFile(string);
    public:
        void Sig_info();        //public methods
        void displayOptions();
        void choiceSelect(int);

        Signal();           //constructors and destructors
        Signal(int);
        Signal(string);
        ~Signal();
};

Signal::Signal(){
    Length=0;
    average=0;
    Max=0;
}
```

```
Signal::Signal(int inputFile){
    char* filename= new char[20];

    if(inputFile<10){
        sprintf(filename,"Raw_data_0%d.txt",inputFile);
    }
    else{
        sprintf(filename,"Raw_data_%d.txt",inputFile);
    }
    readFile(filename);

    delete[] filename;
}

Signal::Signal(string filename){
    readFile(filename);
}

Signal::~Signal(){
    //std::cout << "\nGoodbye Signal";
}

void Signal::operator+(double offset){
    int x=0;
    int count=Length;

    while (count>0) //add the value to each member of data
    {
        alteredData[x]+=offset;
        x++;
        count--;
    }
    alteredMax+=offset; //max will increase by offset
    alteredAverage+=offset; //average will increase by offset
}

void Signal::operator*(double scale){
    int x=0;
    int count=Length;

    while (count>0) //multiply each member of data by scale
    {
        alteredData[x]=signalData[x]*scale;
        x++;
        count--;
    }
    alteredMax*=scale;  //update stats
    alteredAverage*=scale;
}

int Signal::readFile(string filename){
    FILE *fp;
```

```
    fp=fopen(filename.data(),"r");

    if(fp==NULL)    //check for valid file
    {
        std::cout << std::endl << filename << "could not be accessed\n";
        return 1;
    }

    fscanf(fp,"%d %d",&Length,&Max);    //scan first line
    int tempCount=Length;

    int x=0;
    int num;

    while(tempCount>0)
    /*Loop through and add each number in file to vector
        and do same with altered data*/
    {
        fscanf(fp,"%d", &num);
        signalData.push_back(num);
        alteredData.push_back((double)num);
        x++;
        tempCount--;
    }
    fclose(fp);

    mean();     //add stats
    alteredAverage=average;
    maximum();
    alteredMax=Max;
    return 0;
}

void Signal::mean(){
    int total=0;
    int tempCount=Length;

    while(tempCount>0)  //add all vector members
    {
        total+=alteredData[Length-tempCount];
        tempCount--;
    }
    average= (double) total/Length;
}

void Signal::maximum(){
    int total=0;
    int tempCount=Length;

    Max=signalData[0];
    while(tempCount>0)  //compare each and save Max
    {
        Max=(Max>signalData[Length-tempCount])?Max:signalData[Length-tempCount];
```

```
        tempCount--;
    }
}

void Signal::dataReset(){
    int x=0;
    int count=Length;

    while (count>0) //go through and store original data in altered
    {
        alteredData[x]=signalData[x];
        x++;
        count--;
    }

    alteredMax=Max;     //reset stats
    alteredAverage=average;
}

void Signal::Sig_info(){
    std::cout   << "\nLength: " << Length
                << "\nMaximum: " << alteredMax
                << "\nAverage: " << alteredAverage << std::endl;
}

void Signal::displayOptions(){
    cout    <<  "\n1) Offset Data"
            <<  "\n2) Scale Data"
            <<  "\n3) Center Data"
            <<  "\n4) Normalize Data"
            <<  "\n5) Save Data to file"
            <<  "\n6) Display Data"
            <<  "\n7) Reset to original data"
            <<  "\n8) Display These Options"
            <<  "\n9) Exit"
            <<  endl;
}

void Signal::choiceSelect(int choice){
    switch(choice){
        case 1:
            cout    <<  "\nOffset Value: ";
            cin     >>  offsetVal;
            *this + offsetVal;
            break;
        case 2:
            cout    <<  "\nScale Value: ";
            cin     >>  scaleVal;
            *this * scaleVal;
            break;
        case 3:
            centerFile();
            break;
```

```
            case 4:
                normalizeFile();
                break;
            case 5:
                {
                string filename;
                cout    <<  "\nFilename to save to: ";
                cin     >>  filename;
                saveFile(filename);
                }
                break;
            case 6:
                Sig_info();
                break;
            case 7:
                dataReset();
                break;
            case 8:
                displayOptions();
                break;
            case 9:
                dataReset();
                break;
            default:
                cout    <<  "\nInvalid choice. Try again: ";
        }
    }

    void Signal::saveFile(string filename){
        FILE *write;
        write=fopen(filename.data(),"w");

        fprintf(write,"%d %f\n",Length, alteredMax);    //print first 2 stats

        int x=0;
        int tempCount=Length;
        while(tempCount>0)  //print each member into file
        {
            fprintf(write,"%f\n",alteredData[x]);
            x++;
            tempCount--;
        }

        fclose(write);

        std::cout << std::endl << filename << " has been saved\n";
    }

    Signal operator+(const Signal &lhs, const Signal &rhs){
        Signal sum=lhs;
        int temp=lhs.Length;
        if(lhs.Length==rhs.Length){
            while(temp>0){
```

```
            sum.signalData[temp-1]=lhs.signalData[temp-1]+rhs.signalData[temp-1];
            sum.alteredData[temp-1]=sum.signalData[temp-1];
            temp--;
        }
        sum.Max=(lhs.alteredMax>rhs.alteredMax)? lhs.alteredMax : rhs.alteredMax;
        sum.alteredMax=sum.Max;
        sum.average=(lhs.alteredAverage+rhs.alteredAverage)/2;
        sum.alteredAverage=sum.average;
    }
    else
        cout<<"Different Lengths of Data cannot be added";
    return sum;
}

int main(int argc, char** argv){
    if (argc!=3){   //show help and quit
        cout    <<  "\nUsage:"
                <<  "\nprogramName\t-n\tfileNumber"
                <<  "\nprogramName\t-f\tfileName";
        return 1;
    }
    string flag=argv[1];
    string file = "-f", number = "-n";
    Signal dataSample;
    if (flag==file){    //check for filename flag
        Signal dataFile(argv[2]);
        dataSample=dataFile;
    }
    else if (flag==number){     //check for filenumber flag
        Signal dataNumber(atoi(argv[2]));
        dataSample=dataNumber;
    }
    else{   //otherwise show help and quit
        cout    <<  "\nUsage:"
                <<  "\nprogramName\t-n\tfileNumber"
                <<  "\nprogramName\t-f\tfileName";
        return 1;
    }
    int choice=0;
    dataSample.displayOptions();
    while (choice != 9){    //continuously get choices until exit
        cout    <<  "Choice: ";
        cin     >>  choice;
        dataSample.choiceSelect(choice);
    }

    std::cout << "\n";
    Signal dataSample2= dataSample;
    dataSample2+10;
    Signal newSample=operator+(dataSample, dataSample2);

    dataSample.Sig_info();
    dataSample2.Sig_info();
```

```
        newSample.Sig_info();

        return 0;
    }
```