```
//Thomas Gibbons
//Oct 9, 2016
#include <iostream>
#include <sstream>
#include <cstring>
#include <stdio.h>      //sprintf
#include <stdlib.h>     //atoi

class Signal{
    private:
        int* signalData;
        int Max;
        int Length;
        double average;
        void mean();
        int readFile(char*);
        double* alteredData;
        double alteredMax;
        double alteredAverage;
    public:
        int copyFile(char* sourceFile, char* targetFile);
        void offsetFile(double);
        void scaleFile(double);
        void centerFile(){offsetFile(average*(-1));}
        void normalizeFile(){scaleFile(1.0/Max);}
        void saveFile(char*);
        void Sig_info();
        Signal();
        Signal(int);
        Signal(char*);
        ~Signal();
};

Signal::Signal(){
    std::cout << "\nNo File number or name given";
    signalData=NULL;
    alteredData=NULL;
    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);
//std::cout << filename;
//std::cout << "Testing2";
    readFile(filename);
//std::cout << "Testing3";
    delete[] filename;
```

```
}

Signal::Signal(char* filename){
    readFile(filename);
}

Signal::~Signal(){
    if(signalData!=NULL)
        delete[] signalData;
    if(alteredData!=NULL)
        delete[] alteredData;
    std::cout << "\nGoodbye Signal";
}

int Signal::copyFile(char* sourceFile, char* targetFile){
    /*  input:  name of source file
                name of file it should be copied to*/
    FILE *fp1;
    fp1=fopen(sourceFile,"r");
    if(fp1==NULL)
    {
        printf("%s could not be accessed",sourceFile);
        return 1;
    }

    FILE *fp2;
    fp2=fopen(targetFile,"w");
    if(fp2==NULL)
    {
        printf("%s could not be accessed",targetFile);
        return 1;
    }

    char ch;
    while((ch=fgetc(fp1))!=EOF)
        fputc(ch,fp2);

    fclose(fp1);
    fclose(fp2);

    return 0;
}

void Signal::offsetFile(double offset){
/*  input:  value of offset
    output: store alteredData*/
{
    alteredMax=Max;
    alteredAverage=average;

    int x=0;
    int count=Length;
```

```
        while (count>0)
        {
            alteredData[x]=signalData[x]+offset;
            x++;
            count--;
        }
        alteredMax+=offset;
        alteredAverage+=offset;
    }
    }

    void Signal::scaleFile(double scale){
    /*  input:  value of scale
        output: store alteredData*/
    {
        alteredMax=Max;
        alteredAverage=average;

        int x=0;
        int count=Length;

        while (count>0)
        {
            alteredData[x]=signalData[x]*scale;
            x++;
            count--;
        }
        alteredMax*=scale;
        alteredAverage*=scale;
    }
    }

    int Signal::readFile(char* filename){
        FILE *fp;
        fp=fopen(filename,"r");

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

    //std::cout << "1";
        fscanf(fp,"%d %d",&Length,&Max);
        alteredMax=Max;
        int tempCount=Length;
    //std::cout << "2";
        signalData=new int[Length];
        alteredData=new double[Length];
        int x=0;
    //std::cout << "3";
        while(tempCount>0)
        {
```

```
        fscanf(fp,"%d", &signalData[x]);
        alteredData[x]=(double)signalData[x];
        x++;
        tempCount--;
    }
//std::cout << "4";
    fclose(fp);

    mean();
    alteredAverage=average;
    return 0;
}

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

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

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

void Signal::saveFile(char* filename){
    FILE *write;

    write=fopen(filename,"w");

    fprintf(write,"%d %d\n",Length, Max);

    int x=0;
    int tempCount=Length;
    while(tempCount>0)
    {
        fprintf(write,"%d\n",signalData[x]);
        x++;
        tempCount--;
    }

    fclose(write);

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

class handling{
```

```
    public:
        int c;
        int renameLength;
        double offsetVal;
        double scaleVal;
        int nn;
        int oo;
        int ss;
        int SS;
        int CC;
        int NN;
        int rr;
        int hh;
        handling();
        ~handling();
        int handlingArgs(int argc, char** argv);
        void display();
        int inputFile;
};

handling::handling(){
    //define variables and flags
        inputFile=-1;
        renameLength=0;
        offsetVal=0;
        scaleVal=0;
        c=1;
        nn=0,oo=0,ss=0,SS=0,CC=0,NN=0,rr=0,hh=0;
}

handling::~handling(){

}

void handling::display(){
    std::cout   << "\ninputFile="       << inputFile
                << "\nrenameLength="    << renameLength
                << "\noffsetVal="       << offsetVal
                << "\nscaleVal="        << scaleVal
                << "\nnnn="             << nn
                << "\noo="              << oo
                << "\nss="              << ss
                << "\nSS="              << SS
                << "\nCC="              << CC
                << "\nNN="              << NN
                << "\nrr="              << rr
                << "\nhh="              << hh;
}

int handling::handlingArgs(int argc, char** argv){
    //if Lab5 is only argument
    if(argc==1)
    {
```

```
        printf("Usage for help:\t\tLab5\t-h\n");
        return 1;
    }
    //go through input arguments
        while(c<argc)
        {
            /*checks the next arguement for file # and
            sets nn flag if  # is there and bumps count
            so value isn't seen as an invalid option*/
            if(strcmp(argv[c],"-n")==0)
            {
                if(c+1>=argc)
                {
                    printf("\nInvalid input: No file selected\n");
                    printf("Usage for help:\t\tLab5\t-h\n");
                    return 1;
                }
                else
                {
                    if(argv[c+1][0]>='0' && argv[c+1][0]<='9')
                    {
                        inputFile=atoi(argv[c+1]);
                        if(inputFile<0 || inputFile>99)
                        {
                            printf("Invalid input: File number must be between 0 and 99");
                            return 1;
                        }
                        nn++;
                        c++;
                    }
                    else
                    {
                        printf("\nInvalid input: No file selected\n");
                        printf("Usage for help:\t\tLab5\t-h\n");
                        return 1;
                    }

                }
            }
            /*checks the next arguement for offset value and
            sets oo flag if  # is there and bumps count
            so value isn't seen as an invalid option*/
            else if(strcmp(argv[c],"-o")==0)
            {
                if(c+1>=argc)
                {
                    printf("\nInvalid input: No offset selected\n");
                }
                else
                {
                    if(argv[c+1][0]>=48 && argv[c+1][0]<=57)
                    {
                        offsetVal=strtod(argv[c+1],NULL);
```

```
            oo++;
            c++;
        }
        else
        {
            printf("\nInvalid input: Offset not specified");
        }
    }
}
/*checks the next arguement for scale value and
sets ss flag if  # is there and bumps count
so value isn't seen as an invalid option*/
else if(strcmp(argv[c],"-s")==0)
{
    if(c+1>=argc)
    {
        printf("\nInvalid input: No scale selected\n");
    }
    else
    {
        if(argv[c+1][0]>=48 && argv[c+1][0]<=57)
        {
            scaleVal=strtod(argv[c+1],NULL);
            ss++;
            c++;
        }
        else
        {
            printf("\nInvalid input: Scale not specified");
        }
    }
}
//checks if argument are there and sets flag accordingly
else if(strcmp(argv[c],"-S")==0)
    SS++;
else if(strcmp(argv[c],"-C")==0)
    CC++;
else if(strcmp(argv[c],"-N")==0)
    NN++;
/*checks the next arguement for scale value and
sets ss flag if  # is there and bumps count
so value isn't seen as an invalid option*/
else if(strcmp(argv[c],"-r")==0)
{
    if(c+1>=argc)
    {
        printf("\nInvalid input: No name change selected\n");
    }
    else
    {
        rr=c+1;
        while(argv[c+1][renameLength]!='\o')
            renameLength++;
```

```
                    c++;
                }
            }
            //checks for argument, displays help, and exits
            else if(strcmp(argv[c],"-h")==0)
            {
                printf("Program can be run with the following options:\n\n\t");
                printf("-n:\tFile number(value needed)\n\t");
                printf("-o:\tOffset value(value needed)\n\t");
                printf("-s:\tScale factor(value needed)\n\t");
                printf("-S:\tGet statistics\n\t");
                printf("-C:\tCenter the signal\n\t");
                printf("-N:\tNormalize the signal\n\t");
                printf("-r:\tRename files(name needed)\n\t");
                printf("-h:\tHelp\n\n");
                hh++;
                return 1;
            }
            //other options are considered invalid
            else
            {
                printf("\nOption %s is not valid",argv[c]);
            }
            //bump count to next arguement
            c++;
        }
    //file number necessary to continue so terminate if it wasn't found
    if(nn==0)
    {
        printf("\nInvalid input: File number not included\n");
        printf("Usage for help:\t\tLab5\t-h\n");
        return 1;
    }
    //something to do needed to do or terminate
    if((rr+oo+ss+CC+NN)<1)
    {
        printf("\nInvalid input: No tasks to perform\n");
        printf("Usage for help:\t\tLab5\t-h\n");
        return 1;
    }
    printf("\n");
    return 0;
}

int main(int argc, char** argv)
{
    handling arguments;
    int x=arguments.handlingArgs(argc,argv);
    //arguments.display();
    if(x==1)
        return 1;   //terminate due to input error

    Signal dataSample(arguments.inputFile);
```

```
int choice=0;
//creates string of filename user selects
//{
    char* filename=new char[15];
    if(arguments.inputFile<10)
        sprintf(filename,"Raw_data_0%d.txt",arguments.inputFile);
    else
        sprintf(filename,"Raw_data_%d.txt",arguments.inputFile);
//}

if(arguments.rr>0)
//copy raw data to new file
{
    char* newName=new char[arguments.renameLength+4];
    sprintf(newName,"%s.txt",argv[arguments.rr]);

    dataSample.copyFile(filename,newName);
    printf("\n%s copied to %s\n",filename,newName);
    delete[] newName;
}
delete[] filename;

//stats data output
if(arguments.SS>0)
{
    std::cout   << "\n------------------------------"
                << "\nStatistics of original data";
    dataSample.Sig_info();
}

if(arguments.oo>0)
//operations for offsetting data
{
    if(arguments.rr>0)
    {
    //same amount of space for both scaled and offset
        char* outFile11=new char[arguments.renameLength+11];

    //creates string of output file name user selected
        dataSample.offsetFile(arguments.offsetVal);
        sprintf(outFile11,"%s_Offset.txt",argv[arguments.rr]);

    //print offsetted data to output file
        std::cout   << "\n------------------------------"
                    << "\nStatistics of offset data";
        dataSample.Sig_info();
        std::cout << "\nWould you like to save this data?\n1)Yes\n2)No\n";
        std::cin >> choice;
        if(choice==1)
            dataSample.saveFile(outFile11);

    //free memory allocated
```

```
            delete[] outFile11;
        }
        else
        {
        //same amount of space for both scaled and offset
            char* outFile12=new char[18];

        //creates string of output file name user selected
            dataSample.offsetFile(arguments.offsetVal);
            if(arguments.inputFile<10)
                sprintf(outFile12,"Offset_data_0%d.txt",arguments.inputFile);
            else
                sprintf(outFile12,"Offset_data_%d.txt",arguments.inputFile);

        //print offsetted data to output file
            std::cout   << "\n------------------------------"
                        << "\nStatistics of offset data";
            dataSample.Sig_info();
            std::cout << "\nWould you like to save this data?\n1)Yes\n2)No\n";
            std::cin >> choice;
            if(choice==1)
                dataSample.saveFile(outFile12);

        //free memory allocated
            delete[] outFile12;
        }
    }

    if(arguments.ss>0)
    //operations for scaling data
    {
        if(arguments.rr>0)
        {
        //same amount of space for both scaled and offset
            char* outFile21=new char[arguments.renameLength+11];

        //creates string of output file name user selected
            dataSample.scaleFile(arguments.scaleVal);
            sprintf(outFile21,"%s_Scaled.txt",argv[arguments.rr]);

        //print scaled data to output file
            std::cout   << "\n------------------------------"
                        << "\nStatistics of scaled data";
            dataSample.Sig_info();
            std::cout << "\nWould you like to save this data?\n1)Yes\n2)No\n";
            std::cin >> choice;
            if(choice==1)
                dataSample.saveFile(outFile21);
        //free memory allocated
            delete[] outFile21;
        }
        else
        {
```

```
        //same amount of space for both scaled and offset
            char* outFile22=new char[18];

        //creates string of output file name user selected
            dataSample.scaleFile(arguments.scaleVal);
            if(arguments.inputFile<10)
                sprintf(outFile22,"Scaled_data_0%d.txt",arguments.inputFile);
            else
                sprintf(outFile22,"Scaled_data_%d.txt",arguments.inputFile);

        //print scaled data to output file
            std::cout   << "\n------------------------------"
                        << "\nStatistics of scaled data";
            dataSample.Sig_info();
            std::cout << "\nWould you like to save this data?\n1)Yes\n2)No\n";
            std::cin >> choice;
            if(choice==1)
                dataSample.saveFile(outFile22);
        //free memory allocated
            delete[] outFile22;
        }
    }


    //centered data output
    if(arguments.CC>0)
    {
        if(arguments.rr>0)
        {
        dataSample.centerFile();
        char* centeredFile11=new char[arguments.renameLength+13];
        sprintf(centeredFile11,"%s_Centered.txt",argv[arguments.rr]);

        std::cout   << "\n------------------------------"
                        << "\nStatistics of centered data";
            dataSample.Sig_info();
            std::cout << "\nWould you like to save this data?\n1)Yes\n2)No\n";
            std::cin >> choice;
        if(choice==1)
            dataSample.saveFile(centeredFile11);

        delete[] centeredFile11;
        }
        else
        {
        dataSample.centerFile();
        char* centeredFile12=new char[20];
        if(arguments.inputFile<10)
            sprintf(centeredFile12,"Centered_data_0%d.txt",arguments.inputFile);
        else
            sprintf(centeredFile12,"Centered_data_%d.txt",arguments.inputFile);

        std::cout   << "\n------------------------------"
```

```
                    << "\nStatistics of centered data";
            dataSample.Sig_info();
            std::cout << "\nWould you like to save this data?\n1)Yes\n2)No\n";
            std::cin >> choice;
        if(choice==1)
            dataSample.saveFile(centeredFile12);

        delete[] centeredFile12;
        }
    }

    //normalized data output
    if(arguments.NN>0)
    {
        if(arguments.rr>0)
        {
        dataSample.normalizeFile();
        char* normalizedFile11=new char[arguments.renameLength+15];
        sprintf(normalizedFile11,"%s_Normalized.txt",argv[arguments.rr]);

        std::cout   << "\n-------------------------------"
                    << "\nStatistics of normalized data";
            dataSample.Sig_info();
            std::cout << "\nWould you like to save this data?\n1)Yes\n2)No\n";
            std::cin >> choice;
        if(choice==1)
            dataSample.saveFile(normalizedFile11);

        delete[] normalizedFile11;
        }
        else
        {
        dataSample.normalizeFile();
        char* normalizedFile12=new char[22];
        if(arguments.inputFile<10)
            sprintf(normalizedFile12,"Normalized_data_0%d.txt",arguments.inputFile);
        else
            sprintf(normalizedFile12,"Normalized_data_%d.txt",arguments.inputFile);

        std::cout   << "\n-------------------------------"
                    << "\nStatistics of normalized data";
            dataSample.Sig_info();
            std::cout << "\nWould you like to save this data?\n1)Yes\n2)No\n";
            std::cin >> choice;
        if(choice==1)
            dataSample.saveFile(normalizedFile12);

        delete[] normalizedFile12;
        }
    }

    //end succesfully
        std::cout << "\n";
```

```
        return 1;
    }
```