Thomas Gibbons

November 9, 2016

ECE 3220

LAB 9

Objective

The objective of this lab was to become more familiar with strings in C++ as well as learn about vectors and operators. This will make it so there will be little to no need for memory allocation.

Discussion

In this lab we learned to convert our previous lab (lab7) into a program with vectors, strings, and operators. So the first order of business was to take care of the strings. So I used the replace all to change anywhere that said char* to string and then I edited here and there. For example, where memory was allocated, I removed the allocation and deallocation. Also when opening files and need the char*, I used the data() method in string to get it for that purpose.

For the vector part, I changed the signalData from a int* to vector<int>. As well as alteredData to double. This caused me to have to change how I got my data from file. But due to its resemblance of arrays not much else changed.

```
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--;
}
```

Then came the operators. Changing offset and scale was very easy. It was as simple as changing the name. Nothing within the method changed at all. I then changed normalize and center to be used properly.

```
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;
}
```

```
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
}
```

The only problem I encountered was in my non-member operator. I ended up commenting it out so it would compile, but even after defining it as a friend to the Signal class, it would not allow me to access private variables. I needed that to add arrays and use length that were supposed to be kept private.

```
Signal operator+(const Signal &lhs, const Signal &rhs){
    Signal sum;
    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];
        }
        sum.Max=(lhs.Max>rhs.max)?lhs.Max:rhs.Max;
        sum.alteredMax=sum.Max;
        sum.average=(lhs.average+rhs.average)/2;
        sum.alteredAverage=sum.average;
    }
    else
        cout<<"Different Lengths of Data cannot be added";
    return sum;
}
```

In the vectors file that was supplied, I added more vectors to show every initialization method and printed them. I then took in ints and strings from keyboard to a vector with a chance to exit for exercise 3.14 and 3.15. I also included all of the operations for vectors at the end.

```
//Take in ints from cin
//EXERCISE 3.14
vector<int> input;
int num;

cout    << "\nEnter numbers for vector (enter -1 to quit)\n"
        << "? ";
cin     >> num;
while(num!=-1){
    input.push_back(num);
    cout    << "? ";
    cin     >> num;
}
for(auto i:input)   //print input
    cout << i << endl;
```

```
Enter numbers for vector (enter -1 to quit)
? 10
? 17
? 16
? 25
? -1
10
17
16
25
```

```
//Take in strings from cin
//EXERCISE 3.15
vector<string> input2;
string word;
string end="DONE";

cout    << "\nEnter strings for vector (enter DONE to quit)\n"
        << "? ";
cin     >> word;
while(word!=end){
    input2.push_back(word);
    cout    << "? ";
    cin     >> word;
}
for(auto i:input2)  //print input
    cout << i << endl;
```

```
Enter strings for vector (enter DONE to quit)
? Thomas
? Patrick
? Gibbons
? Good
? Grade
? Lab
? DONE
Thomas
Patrick
Gibbons
Good
Grade
Lab
```

For the string file, I played around with all the operations for strings and then wrote a simple part that was meant to take a line and switch the case of all letters and add periods to spaces.

```
    cout << "\nEnter some text, finish it with an &" << endl;
    getline(cin, line, '&');
    cout << line << endl;       //inputed line

    int length=line.size();
    while (length>0){
        if (isspace(line[length]))
            line[length]= '.';
        else if (islower(line[length]))
            line[length]=toupper(line[length]);
        else if (isupper(line[length]))
            line[length]=tolower(line[length]);
        length--;
    }
    cout << line << endl;       //edited line
```

```
Enter some text, finish it with an &
Ethan Nelson smells like burnt CUPCAKES&

Ethan Nelson smells like burnt CUPCAKES

eTHAN.nELSON.SMELLS.LIKE.BURNT.cupcakes
```

For the operators file there was not much to do, but there were some things that said try like when multiply obj1 and obj2 by a constant. It is alright when the constant comes after the object but when you reverse them it will not compile. The multiplication sign has to be on the right of the object.

As far as the actual lab it has the same results as the previous lab. It is the exact same output just with different code using vectors, strings, and operators.

As always a link to my github account is https://github.com/tgibbons95/Lab9.