

Lab5

Generated by Doxygen 1.8.8

Sat Nov 4 2017 01:40:59

Contents

1	readData	2
2	countRows	3
3	overloading	4
4	addtimes	5
5	readtimes	6
6	Specification	7
7	Analysis	8
8	Design	9
9	Test	10
10	Class Index	17
10.1	Class List	17
11	File Index	18

11.1 File List	18
12 Class Documentation	19
12.1 AQIData Struct Reference	19
12.1.1 Member Data Documentation	19
12.2 MERGESORT< AQIData > Class Template Reference	19
12.2.1 Constructor & Destructor Documentation	20
12.2.2 Member Function Documentation	20
12.3 QUICKSORT< AQIData > Class Template Reference	21
12.3.1 Member Function Documentation	21
13 File Documentation	23
13.1 addtimes.cpp File Reference	23
13.1.1 Function Documentation	23
13.2 countRows.cpp File Reference	23
13.2.1 Function Documentation	24
13.3 info.cpp File Reference	24
13.3.1 Function Documentation	25
13.4 lab.h File Reference	25
13.4.1 Function Documentation	26
13.5 main.cpp File Reference	30

13.5.1 Function Documentation	30
13.6 overloading.cpp File Reference	34
13.6.1 Function Documentation	34
13.7 readData.cpp File Reference	35
13.7.1 Function Documentation	36
13.8 readtimes.cpp File Reference	36
13.8.1 Function Documentation	37
13.9 SOMETYPE.hpp File Reference	37
13.9.1 Function Documentation	38
13.10specification.dox File Reference	40

1 readData

read some columns from file *f* and store them in *aqi* array

Parameters

<i>f</i>	filename
<i>aqi</i>	array to put data in
<i>n</i>	number of elements in the array
<i>read</i>	the first string in the file <i>f</i> and return it read the file and store the value into the array of structs we store the county names and the AQI (airquality index of that county)

2 countRows

This program will count the rows in the .csv file. Everytime a new row is read n(the counter) gets incremented till the end

Parameters

<i>f</i>	The string that is the filename
<i>out</i>	we return n the number of rows

3 overloading

This is the operator overloading pages I overloaded the $>$, $<$, $>=$, and $<=$ operators This was done so I could compare the AQI values specifically to one another without having to change the authors code

4 addtimes

This function reads the times it takes for the sorting algorithms to complete and then writes them into a file. that file is called times.txt

Parameters

<i>filename</i>	the filename you want to it to write to
<i>inputstring</i>	the string you want to output

5 readtimes

Void function to read the information in a file and display it No parameters, Called in main. Select Time in html to see the exact output of this function

6 Specification

This is the AQI sorting program. It uses a variety of sorting algorithms to sort the AQI or air quality index of multiple counties from low to high. Sorting algorithms each have different levels of efficiency so our main focus is to test and see which algorithm is the best to use for our data. We use Bubble, Selection, Insertion, Merge, and Quick sorting algorithms. The UI is done on an a page formatted with html.

Features:

- 1) A user can test multiple sorting algorithms.
- 2) The user can check the times each algorithm took.
- 3) UI is very easy and clear to use.

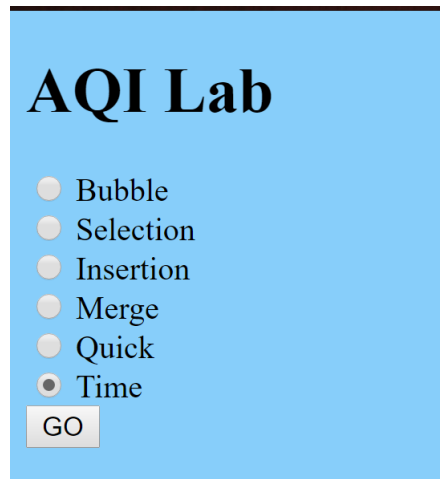
7 Analysis

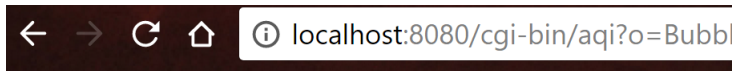
When the user goes to the html page the program is already running. Here the user will be greeted by 6 options to select from. 5 of these options are sorting algorithms the other one is an option to see the times it takes for them to sort. When a sort is selected the page will load the name of the sort and then all the sorted info below it. The fastest sort was quick sort which took less than a second. The slowest was bubble which varied from 40 seconds to 3 minutes.

8 Design

Html was what we used to create the UI for this lab. we were able to have buttons to select what we wanted and customize the page very well. Html is widely used in the industry so adopting this platform was the proper move. The biggest advantage I see is that I am able to edit, change, and add things very easily and I do not need the user to have certain libraries installed for them to see this.

9 Test





Bubble Sort

County: "Mobile"	AQI Data: 0
County: "Morgan"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0
County: "Shelby"	AQI Data: 0

← → ↻ 🏠 ⓘ localhost:8080/cgi-bin/aqi?o=Inser

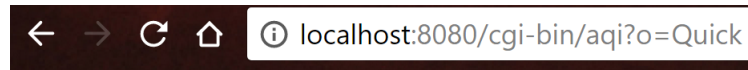
Insertion

County:	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Oneida"	AQI Data: 0
County: "Wood"	AQI Data: 0
County: "Wood"	AQI Data: 0
County: "Monongalia"	AQI Data: 0

← → ↻ 🏠 ⓘ localhost:8080/cgi-bin/aqi?o=Merge

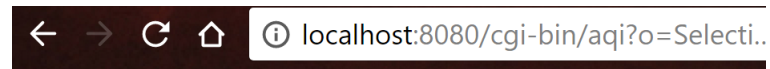
Merge Sort

County: "Macoupin"	AQI Data: 0
County: "Morgan"	AQI Data: 0
County: "Tazewell"	AQI Data: 0
County: "Whatcom"	AQI Data: 0
County: "Monongalia"	AQI Data: 0
County: "Beaufort"	AQI Data: 0
County: "Milam"	AQI Data: 0
County: "Hutchinson"	AQI Data: 0
County: "Monongalia"	AQI Data: 0
County: "Monongalia"	AQI Data: 0
County: "Wabash"	AQI Data: 0
County: "Franklin"	AQI Data: 0
County: "Seneca"	AQI Data: 0
County: "St. Charles"	AQI Data: 0
County: "Franklin"	AQI Data: 0
County: "Columbiana"	AQI Data: 0
County: "Titus"	AQI Data: 0
County: "Franklin"	AQI Data: 0
County: "Titus"	AQI Data: 0



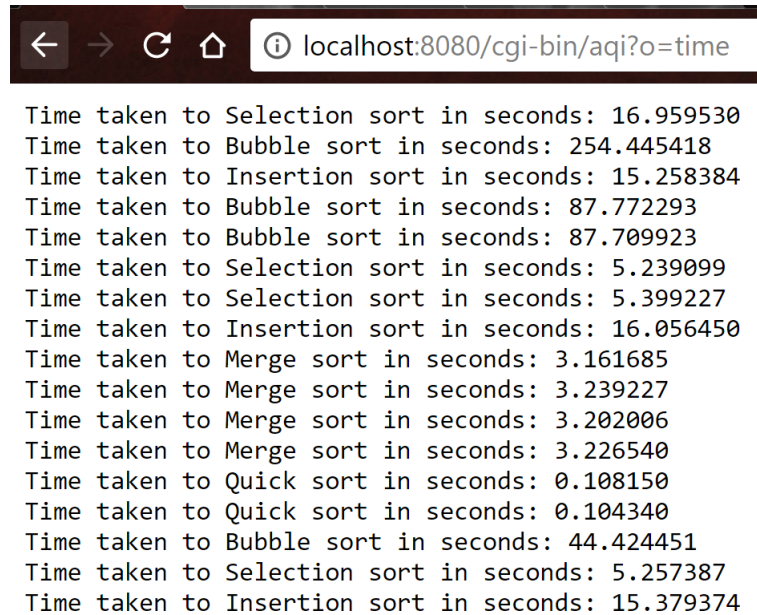
Quick Sort

County: "Morgan"	AQI Data: 0
County: "Pike"	AQI Data: 0
County: "Mayes"	AQI Data: 0
County: "Mason"	AQI Data: 0
County: "Pike"	AQI Data: 0
County: "Pike"	AQI Data: 0
County: "Mayes"	AQI Data: 0
County: "Pike"	AQI Data: 0
County: "Pike"	AQI Data: 0
County: "Pike"	AQI Data: 0
County: "Pike"	AQI Data: 0
County: "Pike"	AQI Data: 0
County: "Pike"	AQI Data: 0
County: "Mason"	AQI Data: 0
County: "Pike"	AQI Data: 0
County: "La Salle"	AQI Data: 0
County: "Mason"	AQI Data: 0
County: "Mason"	AQI Data: 0
County: "Mason"	AQI Data: 0
County:	AQI Data: 0
County: "Mason"	AQI Data: 0
County: "La Salle"	AQI Data: 0
County: "Mason"	AQI Data: 0
County: "Woodbury"	AQI Data: 0



Selection

County: "Seneca"	AQI Data: 0
County: "Woodbury"	AQI Data: 0
County: "Haywood"	AQI Data: 0
County: "St. Charles"	AQI Data: 0
County: "Saint Charles"	AQI Data: 0
County: "Macoupin"	AQI Data: 0
County: "Beaufort"	AQI Data: 0
County: "Franklin"	AQI Data: 0
County: "Franklin"	AQI Data: 0
County: "Franklin"	AQI Data: 0
County: "Martin"	AQI Data: 0
County: "Mason"	AQI Data: 0
County: "La Salle"	AQI Data: 0
County: "Franklin"	AQI Data: 0
County: "La Salle"	AQI Data: 0
County: "Monongalia"	AQI Data: 0
County: "Milam"	AQI Data: 0
County: "Mitchell"	AQI Data: 0
County: "Woodbury"	AQI Data: 0
County: "Franklin"	AQI Data: 0
County: "St. Lawrence"	AQI Data: 0
County: "Franklin"	AQI Data: 0
County: "Hutchinson"	AQI Data: 0
County: "Martin"	AQI Data: 0
County: "Titus"	AQI Data: 0
County: "Jessamine"	AQI Data: 0
County: "Columbiana"	AQI Data: 0
County: "Beaufort"	AQI Data: 0



The image shows a screenshot of a web browser window. The address bar at the top displays the URL "localhost:8080/cgi-bin/aqi?o=time". Below the address bar, there is a list of 15 lines of text, each representing the time taken for a specific sorting algorithm. The times are listed in seconds. The algorithms and their corresponding times are: Selection sort (16.959530s), Bubble sort (254.445418s), Insertion sort (15.258384s), Bubble sort (87.772293s), Bubble sort (87.709923s), Selection sort (5.239099s), Selection sort (5.399227s), Insertion sort (16.056450s), Merge sort (3.161685s), Merge sort (3.239227s), Merge sort (3.202006s), Merge sort (3.226540s), Quick sort (0.108150s), Quick sort (0.104340s), Bubble sort (44.424451s), Selection sort (5.257387s), and Insertion sort (15.379374s).

```
Time taken to Selection sort in seconds: 16.959530
Time taken to Bubble sort in seconds: 254.445418
Time taken to Insertion sort in seconds: 15.258384
Time taken to Bubble sort in seconds: 87.772293
Time taken to Bubble sort in seconds: 87.709923
Time taken to Selection sort in seconds: 5.239099
Time taken to Selection sort in seconds: 5.399227
Time taken to Insertion sort in seconds: 16.056450
Time taken to Merge sort in seconds: 3.161685
Time taken to Merge sort in seconds: 3.239227
Time taken to Merge sort in seconds: 3.202006
Time taken to Merge sort in seconds: 3.226540
Time taken to Quick sort in seconds: 0.108150
Time taken to Quick sort in seconds: 0.104340
Time taken to Bubble sort in seconds: 44.424451
Time taken to Selection sort in seconds: 5.257387
Time taken to Insertion sort in seconds: 15.379374
```

10 Class Index

10.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AQIData	19
MERGESORT< AQIData >	19
QUICKSORT< AQIData >	21

11 File Index

11.1 File List

Here is a list of all files with brief descriptions:

addtimes.cpp	23
countRows.cpp	23
info.cpp	24
lab.h	25
main.cpp	30
overloading.cpp	34
readData.cpp	35
readtimes.cpp	36
SOMETYPE.hpp	37

12 Class Documentation

12.1 AQIData Struct Reference

```
#include <lab.h>
```

Public Attributes

- string [county](#)
- int [AQI](#)

12.1.1 Member Data Documentation

12.1.1.1 int [AQIData::AQI](#)

12.1.1.2 string [AQIData::county](#)

The documentation for this struct was generated from the following file:

- [lab.h](#)

12.2 MERGESORT< AQIData > Class Template Reference

```
#include <lab.h>
```

Public Member Functions

- [MERGESORT](#) (int n)
- [~MERGESORT](#) ()
- void [Sort](#) ([AQIData](#) a[], int n)

12.2.1 Constructor & Destructor Documentation

12.2.1.1 `template<class AQIData> MERGESORT< AQIData >::MERGESORT (int n) [inline]`

```
51 {work = new AQIData[n]; }
```

12.2.1.2 `template<class AQIData> MERGESORT< AQIData >::~~MERGESORT () [inline]`

```
53 {delete [] work;}
```

12.2.2 Member Function Documentation

12.2.2.1 `template<class AQIData > void MERGESORT< AQIData >::Sort (AQIData a[], int n)`

```
78 {
79     int n1, n2;
80     AQIData *a2;
81     if (n <= 2) { //Base Case
82         if ( n== 2 && a[1] < a[0])
83             Swap(a[0], a[1]);
84     }
85     else { // Recursive case:
86         n1 = n/2; n2 = n - n1;
```

```
87         a2 = &a[n1];
88
89         Sort(a, n1);
90         Sort(a2, n2);
91         Merge(a, n1, a2, n2);
92     }
93 }
```

The documentation for this class was generated from the following files:

- [lab.h](#)
- [SOMETYPE.hpp](#)

12.3 QUICKSORT< AQIData > Class Template Reference

```
#include <lab.h>
```

Public Member Functions

- void [Sort](#) ([AQIData](#) a[], int n)

12.3.1 Member Function Documentation

12.3.1.1 `template<class AQIData> void QUICKSORT< AQIData >::Sort (AQIData a[], int n)`

```
163 {
164     int p;
165 }
```



```
166     if (n <=2) {
167         if (n ==2 && a[1] < a[0])
168             Swap(a[0], a[1]);
169     }
170     else {
171         p = Split(a, n);
172         Sort(a, p);
173         Sort(&a[p+1], n-p-1);
174     }
175 }
```

The documentation for this class was generated from the following files:

- [lab.h](#)
- [SOMETYPE.hpp](#)

13 File Documentation

13.1 addtimes.cpp File Reference

```
#include "lab.h"
```

Functions

- bool `addtimes` (string filename, string inputstring)

13.1.1 Function Documentation

13.1.1.1 bool addtimes (string filename, string inputstring)

```
3 {  
4  
5     ofstream inpfiler(filename, ios::app);  
6     inpfiler << inputstring << endl;  
7     inpfiler.close();  
8     return true;  
9 }
```

13.2 countRows.cpp File Reference

```
#include "lab.h"
```

Functions

- int `countRows` (string f)

13.2.1 Function Documentation

13.2.1.1 int countRows (string f)

```
3 {  
4     std::ifstream ifs(f.c_str());  
5     int n = 0;  
6     string s;  
7     while(getline(ifs,s)) n++;  
8  
9  
10    return n;  
11 }
```

13.3 info.cpp File Reference

```
#include "lab.h"
```

Functions

- int `average` ()

13.3.1 Function Documentation

13.3.1.1 int average ()

```
4 {  
5  
6  
7  
8 }
```

13.4 lab.h File Reference

```
#include <iostream>  
#include <stdlib.h>  
#include <stdio.h>  
#include <fstream>  
#include <chrono>  
#include <ctime>  
#include <iomanip>  
#include <sstream>  
#include <string>
```

Classes

- struct [AQIData](#)
- class [MERGESORT< AQIData >](#)
- class [QUICKSORT< AQIData >](#)

Functions

- void `readData` (string f, `AQIData` aqi[], int n)
- int `countRows` (string f)
- bool `operator>=` (const `AQIData` &lhs, const `AQIData` &rhs)
- bool `operator<=` (const `AQIData` &lhs, const `AQIData` &rhs)
- bool `operator>` (const `AQIData` &lhs, const `AQIData` &rhs)
- bool `operator<` (const `AQIData` &lhs, const `AQIData` &rhs)
- template<class `AQIData` >
void `Swap` (`AQIData` &a, `AQIData` &b)
- template<class `AQIData` >
void `BubbleSort` (`AQIData` a[], int n)
- bool `addtimes` (string filename, string inputstring)
- void `readtimes` ()

13.4.1 Function Documentation

13.4.1.1 bool `addtimes` (string *filename*, string *inputstring*)

```
3 {  
4  
5     ofstream inpfiler(filename, ios::app);  
6     inpfiler << inputstring << endl;  
7     inpfiler.close();  
8     return true;  
9 }
```

13.4.1.2 template<class AQIData > void BubbleSort (AQIData a[], int n)

```
55 {  
56     int i, disorder = n;  
57  
58     while (disorder) {  
59  
60         disorder = 0;  
61  
62         for (i = 1; i < n; i++) {  
63             if (a[i] < a[i-1]) {  
64  
65                 Swap(a[i], a[i-1]);  
66                 disorder++;  
67  
68             }  
69         }  
70         n--;  
71     }  
72 }
```

13.4.1.3 int countRows (string f)

```
3 {  
4     std::ifstream ifs(f.c_str());  
5     int n = 0;  
6     string s;  
7     while(getline(ifs,s)) n++;  
8  
9  
10    return n;  
11 }
```

13.4.1.4 bool operator< (const AQIData & lhs, const AQIData & rhs)

```
24 {  
25     return(lhs.AQI < rhs.AQI);  
26 }
```

13.4.1.5 bool operator<= (const AQIData & lhs, const AQIData & rhs)

```
14 {  
15     return(lhs.AQI <= rhs.AQI);  
16 }
```

13.4.1.6 bool operator> (const AQIData & lhs, const AQIData & rhs)

```
19 {  
20     return(lhs.AQI > rhs.AQI);  
21 }
```

13.4.1.7 bool operator>= (const AQIData & lhs, const AQIData & rhs)

This is the operator overloading pages I overloaded the >, <, >=, and <= operators This was done so I could compare the AQI values specifically to one another without having to change the authors code

```
9 {  
10     return(lhs.AQI >= rhs.AQI);  
11 }
```

13.4.1.8 void readData (string f, AQIData aqi[], int n)

```
10 {
```

```
11     ifstream ifs (f.c_str());
12     string s; char comma = ',';
13     getline(ifs,s);
14     for( int i = 0; i < n ; i++) {
15         getline(ifs,s,','); //read and ignore the state
16         getline(ifs, aqi[i].county, ',');
17         getline(ifs,s,',');
18         getline(ifs,s,','); //read and ignore state
19         getline (ifs,s,','); //read and ignore the Days of AQI
20         ifs >> aqi[i].AQI >> comma;
21         getline(ifs,s);
22     }
23     ifs.close();
24 }
```

13.4.1.9 void readtimes ()

```
3 {
4     string line;
5     ifstream inpf("times.txt");
6     if (inpf.is_open())
7     {
8         while (getline (inpf, line) )
9         {
10             cout << line << endl;
11         }
12     }
13     inpf.close();
14
15 }
```


13.4.1.10 `template<class AQIData > void Swap (AQIData & a, AQIData & b)`

```
4 {  
5     AQIData temp = a;  
6     a = b;  
7     b = temp;  
8 }
```

13.5 main.cpp File Reference

```
#include "lab.h"  
#include "SOMETYPE.hpp"
```

Functions

- void `list` ()
- int `main` ()

13.5.1 Function Documentation

13.5.1.1 `void list ()`

This is the main function. It handles most of the control Every If statement is a different sort It takes data from the html file and then and then choses which sorting algorithm to use. This also reads and writes to a file called times.txt This stores the times each sort took so we can compare the n^2 and $n \log n$ sort. We use Bubble, Selection, Insertion for n^2 and Merge and Quick sort for $n \log n$.

13.5.1.2 int main()

```
17 {
18     string s = getenv("QUERY_STRING");
19     string f = "/home/debian/data/aqi.csv";
20     int n = countRows(f);
21     AQIData* aqi = new AQIData[n];
22
23     MERGESORT<AQIData> mergesort(n);
24     QUICKSORT<AQIData> quicksort;
25
26     readData(f,aqi,n);
27
28     if(s == "o=time")
29     {
30         readtimes();
31     }
32     if(s == "o=Bubble")
33     {
34         cout << "Bubble Sort" << endl;
35         auto t1 = std::chrono::high_resolution_clock::now();
36         BubbleSort(aqi, n);
37         auto t2 = std::chrono::high_resolution_clock::now();
38         auto time_span = std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
39         double time = time_span.count();
40         string str = "Time taken to Bubble sort in seconds: " + std::to_string(time);
41         addtimes("times.txt", str);
42         for(int i = 0; i < n; i++)
43         {
44             cout << left << setw(7) << "County: " << left << setw(30) << aqi[i].
county;
45             cout << "AQI Data: "<< aqi[i].AQI << endl;
```

```
46     }
47 }
48 if(s == "o=Selection")
49 {
50     cout << "Selection" << endl;
51     auto t1 = std::chrono::high_resolution_clock::now();
52     SelectionSort(aqi, n);
53     auto t2 = std::chrono::high_resolution_clock::now();
54     auto time_span = std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
55     double time = time_span.count();
56     string str = "Time taken to Selection sort in seconds: " + std::to_string(time);
57     addtimes("times.txt", str);
58     for(int i = 0; i < n; i++)
59     {
60         cout << left << setw(7) << "County: " << left << setw(30) << aqi[i].
county;
61         cout << "AQI Data: "<< aqi[i].AQI << endl;
62     }
63 }
64 if(s == "o=Insertion")
65 {
66     cout << "Insertion" << endl;
67     auto t1 = std::chrono::high_resolution_clock::now();
68     InsertionSort(aqi, n);
69     auto t2 = std::chrono::high_resolution_clock::now();
70     auto time_span = std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
71     double time = time_span.count();
72     string str = "Time taken to Insertion sort in seconds: " + std::to_string(time);
73     addtimes("times.txt", str);
74     for(int i = 0; i < n; i++)
75     {
76         cout << left << setw(7) << "County: " << left << setw(30) << aqi[i].
```

```
    county;
77     cout << "AQI Data: "<< aqi[i].AQI << endl;
78     }
79     }
80     if(s == "o=Merge")
81     {
82         cout << "Merge Sort" << endl;
83         auto t1 = std::chrono::high_resolution_clock::now();
84         mergesort.Sort(aqi, n);
85         auto t2 = std::chrono::high_resolution_clock::now();
86         auto time_span = std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
87         double time = time_span.count();
88         string str = "Time taken to Merge sort in seconds: " + std::to_string(time);
89         addtimes("times.txt", str);
90         for(int i = 0; i < n; i++)
91         {
92             cout << left << setw(7) << "County: " << left << setw(30) << aqi[i].
county;
93             cout << "AQI Data: "<< aqi[i].AQI << endl;
94             }
95         }
96         if(s == "o=Quick")
97         {
98             cout << "Quick Sort" << endl;
99             auto t1 = std::chrono::high_resolution_clock::now();
100             quicksort.Sort(aqi, n);
101             auto t2 = std::chrono::high_resolution_clock::now();
102             auto time_span = std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
103             double time = time_span.count();
104             string str = "Time taken to Quick sort in seconds: " + std::to_string(time);
105             addtimes("times.txt", str);
106             for(int i = 0; i < n; i++)
```

```
107     {
108         cout << left << setw(7) << "County: " << left << setw(30) << aqi[i].
    county;
109         cout << "AQI Data: "<< aqi[i].AQI << endl;
110     }
111 }
112
113     return 0;
114 }
```

13.6 overloading.cpp File Reference

```
#include "lab.h"
```

Functions

- bool `operator>=` (const `AQIData` &lhs, const `AQIData` &rhs)
- bool `operator<=` (const `AQIData` &lhs, const `AQIData` &rhs)
- bool `operator>` (const `AQIData` &lhs, const `AQIData` &rhs)
- bool `operator<` (const `AQIData` &lhs, const `AQIData` &rhs)

13.6.1 Function Documentation

13.6.1.1 bool `operator<` (const `AQIData` & lhs, const `AQIData` & rhs)

```
24 {
25     return (lhs.AQI < rhs.AQI);
26 }
```

13.6.1.2 bool operator<= (const AQIData & lhs, const AQIData & rhs)

```
14 {  
15     return(lhs.AQI <= rhs.AQI);  
16 }
```

13.6.1.3 bool operator> (const AQIData & lhs, const AQIData & rhs)

```
19 {  
20     return(lhs.AQI > rhs.AQI);  
21 }
```

13.6.1.4 bool operator>= (const AQIData & lhs, const AQIData & rhs)

This is the operator overloading pages I overloaded the >, <, >=, and <= operators This was done so I could compare the AQI values specifically to one another without having to change the authors code

```
9 {  
10     return(lhs.AQI >= rhs.AQI);  
11 }
```

13.7 readData.cpp File Reference

```
#include "lab.h"
```

Functions

- void [readData](#) (string f, [AQIData](#) aqi[], int n)

13.7.1 Function Documentation

13.7.1.1 void readData (string *f*, AQIData *aqi*[], int *n*)

```
10 {
11     ifstream ifs (f.C_str());
12     string s; char comma = ',';
13     getline(ifs,s);
14     for( int i = 0; i < n ; i++) {
15         getline(ifs,s,','); //read and ignore the state
16         getline(ifs, aqi[i].county, ',');
17         getline(ifs,s,',');
18         getline(ifs,s,','); //read and ignore state
19         getline (ifs,s,','); //read and ignore the Days of AQI
20         ifs >> aqi[i].AQI >> comma;
21         getline(ifs,s);
22     }
23     ifs.close();
24 }
```

13.8 readtimes.cpp File Reference

```
#include "lab.h"
```

Functions

- void [readtimes](#) ()

13.8.1 Function Documentation

13.8.1.1 void readtimes ()

```
3 {
4     string line;
5     ifstream infile("times.txt");
6     if (infile.is_open())
7     {
8         while (getline (infile, line) )
9         {
10             cout << line << endl;
11         }
12     }
13     infile.close();
14
15 }
```

13.9 SOMETYPE.hpp File Reference

Functions

- template<class AQIData >
void [Swap](#) (AQIData &a, AQIData &b)
- template<class AQIData >
void [SelectionSort](#) (AQIData a[], int n)
- template<class AQIData >
void [InsertionSort](#) (AQIData a[], int n)
- template<class AQIData >
void [BubbleSort](#) (AQIData a[], int n)

13.9.1 Function Documentation

13.9.1.1 `template<class AQIData > void BubbleSort (AQIData a[], int n)`

```
55 {  
56     int i, disorder = n;  
57  
58     while (disorder) {  
59  
60         disorder = 0;  
61  
62         for (i = 1; i < n; i++) {  
63             if (a[i] < a[i-1]) {  
64  
65                 Swap(a[i], a[i-1]);  
66                 disorder++;  
67  
68             }  
69         }  
70         n--;  
71     }  
72 }
```

13.9.1.2 `template<class AQIData > void InsertionSort (AQIData a[], int n)`

```
29 {  
30  
31     int i, j;  
32     AQIData aCurrent;  
33  
34     for (i =1; i < n; i++) {  
35
```

```
36         //Save the current element:
37         aCurrent = a[i];
38
39         //Find location j where it should be inserted
40         //among the first i-1 elements:
41         for (j=0; j < i; j++)
42             if (a[j] >= aCurrent) break;
43
44         for (int k = i-1; k >= j; k--)
45             a[k+1] = a[k];
46
47         //Insert saved element where it belongs:
48         a[j] = aCurrent;
49     }
50 }
```

13.9.1.3 `template<class AQIData > void SelectionSort (AQIData a[], int n)`

```
12 {
13     int i, iMax;
14
15     while ( n > 1) {
16         //Find the largest element:
17         for (iMax = 0, i = 1; i < n; i++)
18             if (a[i] > a[iMax]) iMax = i;
19
20         //Swap it with the last element:
21         Swap(a[iMax], a[n-1]);
22         n--;
23     }
24 }
```

13.9.1.4 `template<class AQIData > void Swap (AQIData & a, AQIData & b)`

```
4 {  
5     AQIData temp = a;  
6     a = b;  
7     b = temp;  
8 }
```

13.10 `specification.dox` File Reference