

# Reference Manual

Generated by Doxygen 1.8.8

Wed Nov 1 2017 00:15:27

## Contents

<b>1</b>	<b>countRows</b>	<b>2</b>
<b>2</b>	<b>readData</b>	<b>3</b>
<b>3</b>	<b>Class Index</b>	<b>4</b>
3.1	Class List . . . . .	4
<b>4</b>	<b>File Index</b>	<b>5</b>
4.1	File List . . . . .	5
<b>5</b>	<b>Class Documentation</b>	<b>6</b>
5.1	AQIData Struct Reference . . . . .	6
5.1.1	Member Data Documentation . . . . .	6
5.2	MERGESORT< SOMETYPE > Class Template Reference . . . . .	6
5.2.1	Constructor & Destructor Documentation . . . . .	7
5.2.2	Member Function Documentation . . . . .	7
5.3	QUICKSORT< SOMETYPE > Class Template Reference . . . . .	8
5.3.1	Member Function Documentation . . . . .	8
<b>6</b>	<b>File Documentation</b>	<b>10</b>

6.1	countRows.cpp File Reference . . . . .	10
6.1.1	Function Documentation . . . . .	11
6.2	lab.h File Reference . . . . .	11
6.2.1	Function Documentation . . . . .	14
6.3	lab.hpp File Reference . . . . .	16
6.3.1	Function Documentation . . . . .	18
6.4	main.cpp File Reference . . . . .	19
6.4.1	Function Documentation . . . . .	21
6.5	Overload.cpp File Reference . . . . .	24
6.5.1	Function Documentation . . . . .	25
6.6	readData.cpp File Reference . . . . .	26
6.6.1	Function Documentation . . . . .	26

## 1 `countRows`

Read the first string in file `f` return it.

## 2 readData

Read the first string in file f return it.

### Parameters

<i>f</i>	filename
<i>aq[i]</i>	is an array to put data in /param n number amount of elements in the array

### Returns

void

## 3 Class Index

### 3.1 Class List

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

<a href="#">AQIData</a>	<a href="#">6</a>
<a href="#">MERGESORT&lt; SOMETYPE &gt;</a>	<a href="#">6</a>
<a href="#">QUICKSORT&lt; SOMETYPE &gt;</a>	<a href="#">8</a>

## 4 File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">countRows.cpp</a>	10
<a href="#">lab.h</a>	11
<a href="#">lab.hpp</a>	16
<a href="#">main.cpp</a>	19
<a href="#">Overload.cpp</a>	24
<a href="#">readData.cpp</a>	26

## 5 Class Documentation

### 5.1 AQIData Struct Reference

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

#### Public Attributes

- `std::string` [county](#)
- `std::string` [AQI](#)

#### 5.1.1 Member Data Documentation

5.1.1.1 `std::string` `AQIData::AQI`

5.1.1.2 `std::string` `AQIData::county`

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

- [lab.h](#)

### 5.2 MERGESORT< SOMETYPE > Class Template Reference

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



### Public Member Functions

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

#### 5.2.1 Constructor & Destructor Documentation

##### 5.2.1.1 `template<class SOMETYPE> MERGESORT< SOMETYPE >::MERGESORT ( int n )` [inline]

```
31 { work = new SOMETYPE[n]; }
```

##### 5.2.1.2 `template<class SOMETYPE> MERGESORT< SOMETYPE >::~~MERGESORT ( )` [inline]

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

#### 5.2.2 Member Function Documentation

##### 5.2.2.1 `template<class SOMETYPE > void MERGESORT< SOMETYPE >::Sort ( SOMETYPE a[], int n )`

```
61 {  
62     {  
63         int n1, n2;  
64         SOMETYPE *a2;  
65         if (n <= 2) { // Base case:  
66             if (n == 2 && a[1] < a[0])  
67                 Swap(a[0], a[1]);  
68         }  
69         else { // Recursive case:
```

```
70         n1 = n/2; n2 = n - n1;
71         a2 = &a[n1]; // a2 points to the second half of the
72         // array.
73         Sort(a, n1);
74         Sort(a2, n2);
75         Merge(a, n1, a2, n2);
76     }
77 }
78 }
```

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

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

### 5.3 QUICKSORT< SOMETYPE > Class Template Reference

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

#### Public Member Functions

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

#### 5.3.1 Member Function Documentation

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

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

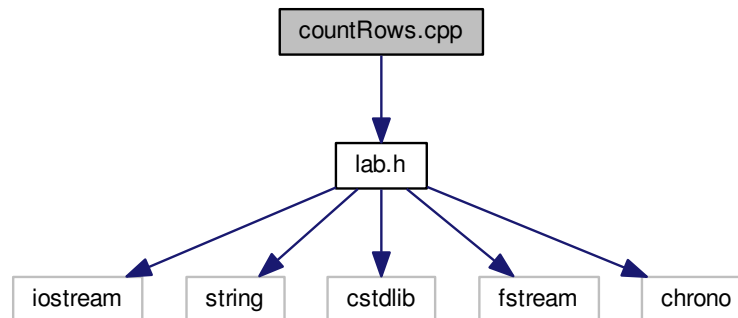
- [lab.h](#)
- [lab.hpp](#)

## 6 File Documentation

### 6.1 countRows.cpp File Reference

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

Include dependency graph for countRows.cpp:



#### Functions

- int `countRows` (std::string f)

### 6.1.1 Function Documentation

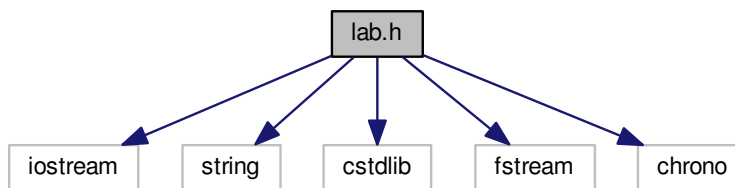
#### 6.1.1.1 int countRows ( std::string f )

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

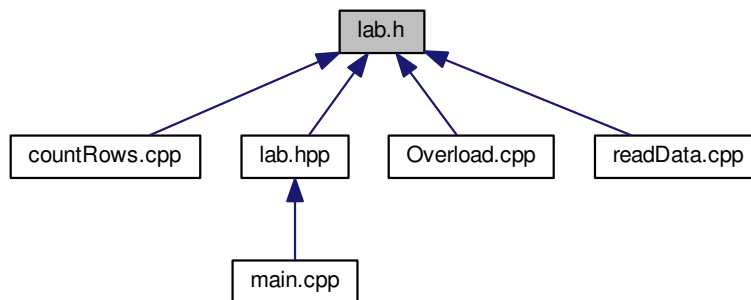
## 6.2 lab.h File Reference

```
#include <iostream>  
#include <string>  
#include <cstdlib>  
#include <fstream>  
#include <chrono>
```

Include dependency graph for lab.h:



This graph shows which files directly or indirectly include this file:



### Classes

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

### Functions

- int [countRows](#) (std::string f)

- void `readData` (`std::string f`, `AQIData aqi[]`, `int n`)
- bool `operator>` (`AQIData &lhs`, `AQIData &rhs`)
- bool `operator<` (`AQIData &lhs`, `AQIData &rhs`)
- bool `operator<=` (`AQIData &lhs`, `AQIData &rhs`)
- bool `operator>=` (`AQIData &lhs`, `AQIData &rhs`)

## 6.2.1 Function Documentation

### 6.2.1.1 int `countRows` ( `std::string f` )

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

### 6.2.1.2 bool `operator<` ( `AQIData &lhs`, `AQIData &rhs` )

```
3 {  
4     return (lhs.AQI < rhs.AQI);  
5 }
```

### 6.2.1.3 bool `operator<=` ( `AQIData &lhs`, `AQIData &rhs` )

```
17 {  
18     return (rhs.AQI <= lhs.AQI);  
19 }
```



**6.2.1.4 bool operator> ( AQIData & lhs, AQIData & rhs )**

```
7 {  
8     return (lhs.AQI > rhs.AQI);  
9 }
```

**6.2.1.5 bool operator>= ( AQIData & lhs, AQIData & rhs )**

```
12 {  
13     return (lhs.AQI >= rhs.AQI);  
14 }
```

**6.2.1.6 void readData ( std::string f, AQIData aqi[], int n )**

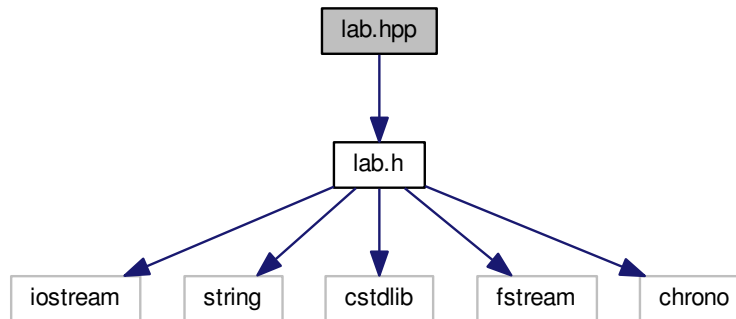
```
4 {  
5  
6     std::ifstream ifs(f.c_str());  
7     std::string s; char comma;  
8     for(int i = 0; i < n; i++)  
9     {  
10         getline(ifs,s,','); //reads state  
11         getline(ifs,aqi[i].county,','); //reads county  
12         getline(ifs,s,','); //ignores county and reads year  
13         getline(ifs,s,',');  
14         getline(ifs,s,',');  
15         getline(ifs,aqi[i].AQI,','); //ignore year and readsDays  
16  
17         //ifs >> aqi[i].AQI >> comma;  
18         getline(ifs,s);  
19     }  
20     //aqi[0].county = s;
```

```
21     ifs.close();  
22  
23  
24 }
```

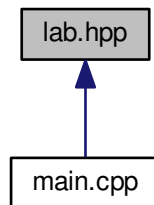
### 6.3 lab.hpp File Reference

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

Include dependency graph for lab.hpp:



This graph shows which files directly or indirectly include this file:



### Functions

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

### 6.3.1 Function Documentation

#### 6.3.1.1 `template<class SOMETYPE > void BubbleSort ( SOMETYPE a[], int n )`

```
42 {  
43     int i, disorder = n;  
44     while(disorder)  
45     {  
46         disorder = 0;  
47         for(i = 1; i < n; i++)  
48         {  
49             if(a[i-1] > a[i])  
50             {  
51                 Swap(a[i], a[i-1]);  
52                 disorder++;  
53             }  
54         }  
55         n--;  
56     }  
57 }
```

#### 6.3.1.2 `template<class SOMETYPE > void InsertionSort ( SOMETYPE a[], int n )`

```
26 {  
27     int i, j;  
28     SOMETYPE aCurrent;  
29     for(i = 1; i < n; i++)  
30     {  
31         aCurrent = a[i];  
32         for(j = 0; j < i; j++)  
33             if(a[j] >= aCurrent) break;  
34         for(int k = i-1; k >= j; k--)
```

```
35         a[k+1] = a[k];
36         a[j] = aCurrent;
37     }
38 }
```

#### 6.3.1.3 template<class SOMETYPE > void SelectionSort ( SOMETYPE a[], int n )

```
13 {
14     int i , iMax;
15     while(n > 1)
16     {
17         for(iMax = 0, i = 1; i < n; i++)
18             if(a[i] > a[iMax]) iMax = i;
19         Swap(a[iMax], a[n-1]);
20         n--;
21     }
22 }
```

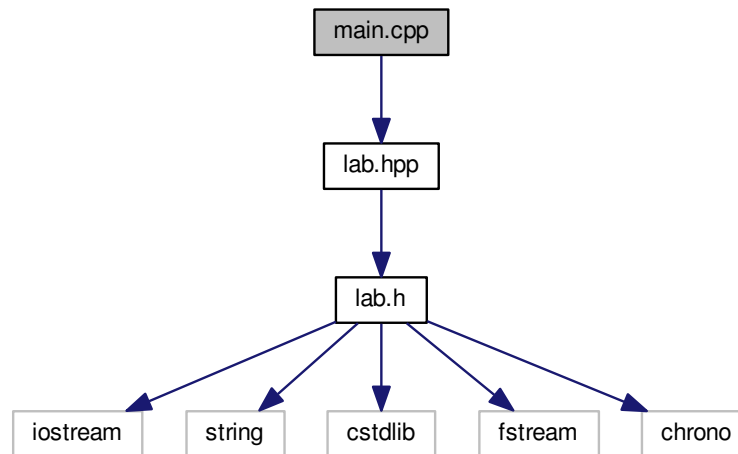
#### 6.3.1.4 template<class SOMETYPE > void Swap ( SOMETYPE & a, SOMETYPE & b ) [inline]

```
5 {
6     SOMETYPE temp = a;
7     a = b;
8     b = temp;
9 }
```

## 6.4 main.cpp File Reference

```
#include "lab.hpp"
```

Include dependency graph for main.cpp:



## Functions

- `int main ()`

## 6.4.1 Function Documentation

## 6.4.1.1 int main ( )

```

for(int i = 0; i < n; i++) { std::cout << "n = " << n << " "; std::cout << aqi[i].county << " "; std::cout <<
aqi[i].AQI << "\n"; }

3 {
4     //std::string s = getenv("QUERY_STRING");
5     //e.g. s may be "o = Bubble"
6     //std::cout << s << std::endl;
7     QUICKSORT<AQIData> qt;
8     MERGESORT<AQIData> ms;
9     std::string f = "/home/debian/data/aqi.csv";
10    int n = countRows(f);
11    AQIData* aqi = new AQIData[n];
12    readData(f, aqi, n);
13    std::string s = getenv("QUERY_STRING");
14    //e.g. s may be "o = Bubble"
15    if(s == "o=Bubble")
16    {
17
18
19        auto t1 = std::chrono::high_resolution_clock::now();
20        BubbleSort(aqi, n);
21        for (int i = 0; i < n; i++)
22            std::cout << "<html>" << aqi[i].county << " " << aqi[i].AQI << "<br></html>";
23        auto t2 = std::chrono::high_resolution_clock::now();
24        auto time_span =
25            std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
26        std::cout << time_span.count() << "\n";
27    }
28    else if(s == "o=Selection")

```

```
34     {
35
36         auto t1 = std::chrono::high_resolution_clock::now();
37         SelectionSort(aqi,n);
38         for (int i = 0; i < n; i++)
39             std::cout << "<html>" << aqi[i].county << " " << aqi[i].AQI<< "<br></html>";
40         auto t2 = std::chrono::high_resolution_clock::now();
41         auto time_span =
42             std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
43         std::cout << time_span.count() << "\n";
44     }
45     else if(s == "o=Insertion")
46     {
47         auto t1 = std::chrono::high_resolution_clock::now();
48         InsertionSort(aqi,n);
49         for (int i = 0; i < n; i++)
50             std::cout << "<html>" << aqi[i].county << " " << aqi[i].AQI<< "<br></html>";
51         auto t2 = std::chrono::high_resolution_clock::now();
52         auto time_span =
53             std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
54         std::cout << time_span.count() << "\n";
55     }
56     else if(s == "o=QuickSort")
57     {
58         auto t1 = std::chrono::high_resolution_clock::now();
59         qt.Sort(aqi,n);
60         for (int i = 0; i < n; i++)
61             std::cout << "<html>" << aqi[i].county << " " << aqi[i].AQI<< "<br></html>";
62         auto t2 = std::chrono::high_resolution_clock::now();
63         auto time_span =
64             std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
65         std::cout << time_span.count() << "\n";
```

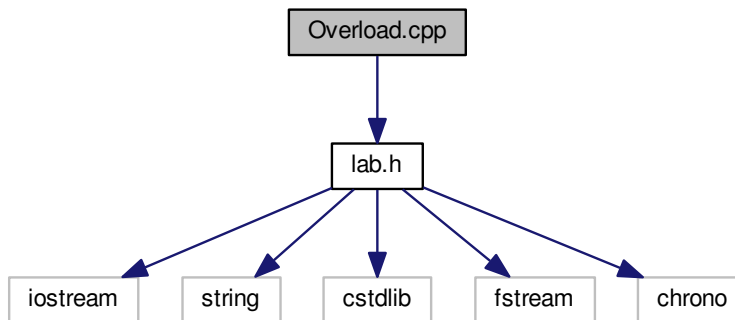


```
66     }
67     else if(s == "o=Merge")
68     {
69         auto t1 = std::chrono::high_resolution_clock::now();
70         ms.Sort(aqi,n);
71         for (int i = 0; i < n; i++)
72             std::cout << "<html>" << aqi[i].county << " " << aqi[i].AQI<< "<br></html>";
73         auto t2 = std::chrono::high_resolution_clock::now();
74         auto time_span =
75             std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
76         std::cout << time_span.count() << "\n";
77     }
78 }
```

## 6.5 Overload.cpp File Reference

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

Include dependency graph for Overload.cpp:



### Functions

- bool `operator<` ([AQIData](#) &lhs, [AQIData](#) &rhs)
- bool `operator>` ([AQIData](#) &lhs, [AQIData](#) &rhs)
- bool `operator>=` ([AQIData](#) &lhs, [AQIData](#) &rhs)
- bool `operator<=` ([AQIData](#) &lhs, [AQIData](#) &rhs)

### 6.5.1 Function Documentation

#### 6.5.1.1 `bool operator< ( AQIData & lhs, AQIData & rhs )`

```
3 {  
4     return (lhs.AQI < rhs.AQI);  
5 }
```

#### 6.5.1.2 `bool operator<= ( AQIData & lhs, AQIData & rhs )`

```
17 {  
18     return (rhs.AQI <= rhs.AQI);  
19 }
```

#### 6.5.1.3 `bool operator> ( AQIData & lhs, AQIData & rhs )`

```
7 {  
8     return (lhs.AQI > rhs.AQI);  
9 }
```

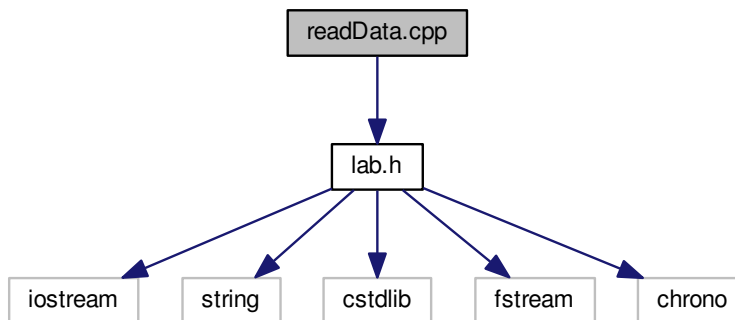
#### 6.5.1.4 `bool operator>= ( AQIData & lhs, AQIData & rhs )`

```
12 {  
13     return (lhs.AQI >= rhs.AQI);  
14 }
```

## 6.6 readData.cpp File Reference

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

Include dependency graph for readData.cpp:



### Functions

- void `readData` (std::string f, `AQIData` aqi[], int n)

#### 6.6.1 Function Documentation

## 6.6.1.1 void readData ( std::string f, AQIData aqi[], int n )

```
4 {
5
6     std::ifstream ifs(f.c_str());
7     std::string s; char comma;
8     for(int i = 0; i < n; i++)
9     {
10         getline(ifs,s,','); //reads state
11         getline(ifs,aqi[i].county,','); //reads county
12         getline(ifs,s,','); //ignores county and reads year
13         getline(ifs,s,',');
14         getline(ifs,s,',');
15         getline(ifs,aqi[i].AQI,','); //ignore year and readsDays
16
17         //ifs >> aqi[i].AQI >> comma;
18         getline(ifs,s);
19     }
20     //aqi[0].county = s;
21     ifs.close();
22
23
24 }
```