Reference Manual

Generated by Doxygen 1.8.8

Wed Nov 1 2017 00:15:27

Contents

1 countRows

Read the first string in file f return it.

2 readData

Read the first string in file f return it.

Parameters

f	filename
aqi[ ]	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:

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

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]

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

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

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

5.2.2 Member Function Documentation

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

```
{
    {
    int n1, n2;
    SOMETYPE *a2;
    if (n <= 2) { // Base case:
        if (n == 2 && a[1] < a[0])
            Swap(a[0], a[1]);
    }
    else { // Recursive case:
        n1 = n/2; n2 = n - n1;
        a2 = &a[n1]; // a2 points to the second half of the
        // array.
        Sort(a, n1);
        Sort(a2, n2);
        Merge(a, n1, a2, n2);
    }
}
}
```

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:

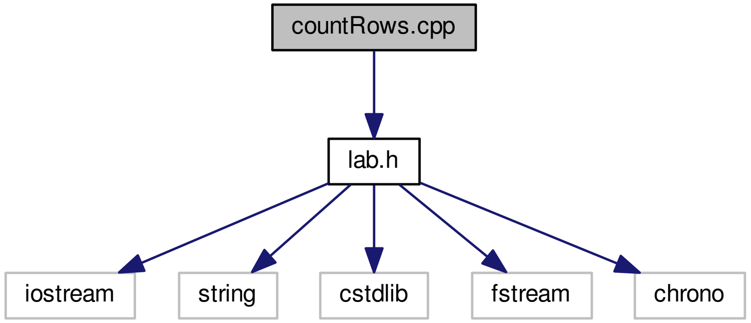
- lab.h
- lab.hpp

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

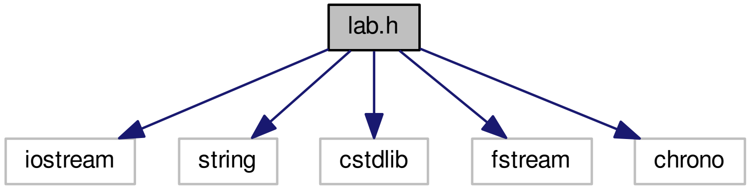
6.1.1.1 int countRows (std::string *f*)

```
{
    std::ifstream ifs(f.c_str());
    int n = 0;
    std::string s;
    while(getline(ifs,s)) n++;
    ifs.close();
    return n;
}
```

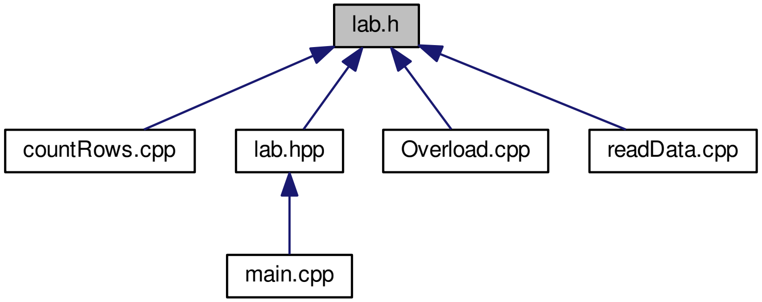
6.2 lab.h File Reference

```
#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*)

```
{
    std::ifstream ifs(f.c_str());
    int n = 0;
    std::string s;
    while(getline(ifs,s)) n++;
    ifs.close();
    return n;
}
```

6.2.1.2 bool operator< (AQIData & *lhs*, AQIData & *rhs*)

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

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

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

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

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

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

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

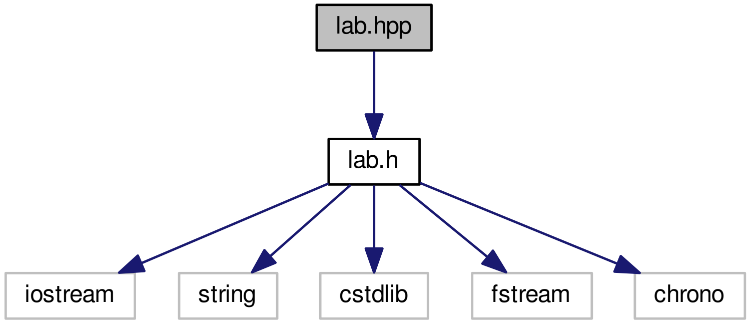
6.2.1.6 void readData (std::string *f*, AQIData *aqi[ ]*, int *n*)

```
{

    std::ifstream ifs(f.c_str());
    std::string s; char comma;
    for(int i = 0; i < n; i++)
    {
        getline(ifs,s,',');//reads state
        getline(ifs,aqi[i].county,',');//reads county
        getline(ifs,s,',');//ignores county and reads year
        getline(ifs,s,',');
        getline(ifs,s,',');
        getline(ifs,aqi[i].AQI,',');//ignore year and readsDays

        //ifs >> aqi[i].AQI >> comma;
        getline(ifs,s);
     }
     //aqi[0].county = s;
    ifs.close();


}
```

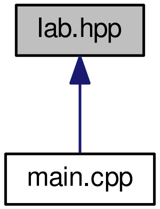
6.3 lab.hpp File Reference

`#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*)

```
{
    int i, disorder = n;
    while(disorder)
    {
        disorder = 0;
        for(i = 1; i < n; i++)
        {
            if(a[i-1] > a[i])
            {
                Swap(a[i], a[i-1]);
                disorder++;
            }
        }
        n--;
    }
}
```

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

```
{
    int i, j;
    SOMETYPE aCurrent;
    for(i = 1; i < n; i++)
    {
        aCurrent = a[i];
        for(j = 0; j < i; j++)
            if(a[j] >=aCurrent) break;
        for(int k = i-1; k>= j; k--)
```

```
            a[k+1] = a[k];
        a[j] = aCurrent;
    }
}
```

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

```
{
    int i , iMax;
    while(n > 1)
    {
        for(iMax = 0, i = 1; i <n; i++)
            if(a[i] > a[iMax]) iMax = i;
        Swap(a[iMax], a[n-1]);
        n--;
    }
}
```

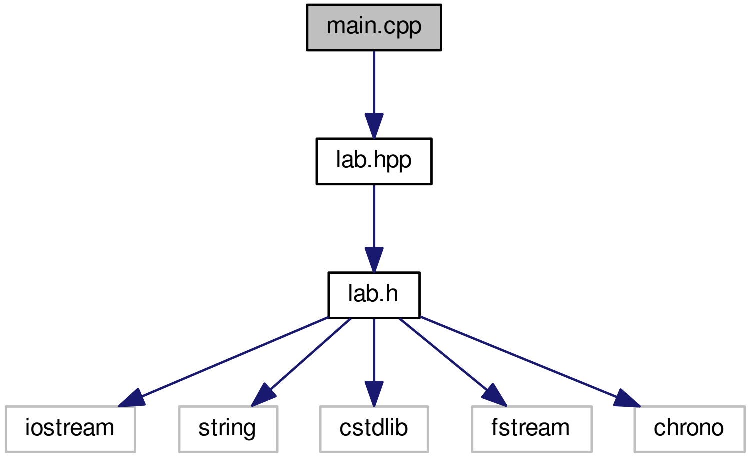
6.3.1.4 template<class SOMETYPE > void Swap (SOMETYPE & *a,* SOMETYPE & *b*) [inline]

```
{
    SOMETYPE temp = a;
    a = b;
    b = temp;
}
```

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 ( )

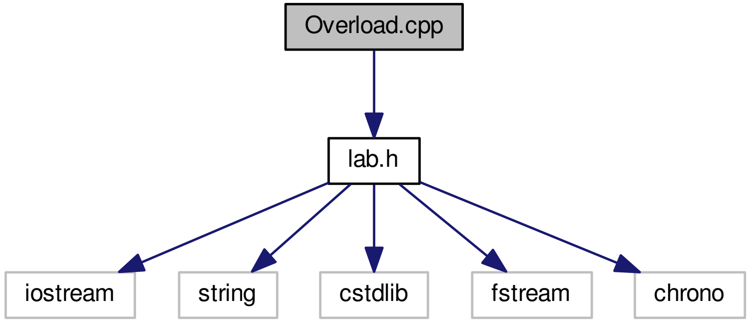
for(int i = 0;i < n; i++) { std::cout << "n = " << n << " "; std::cout << aqi[i].county << " "; std::cout << aqi[i].AQI << "\n"; }

```
{
    //std::string s = getenv("QUERY_STRING");
    //e.g. s may be "o = Bubble"
    //std::cout << s << std::endl;
    QUICKSORT<AQIData> qt;
    MERGESORT<AQIData> ms;
    std::string f = "/home/debian/data/aqi.csv";
     int n = countRows(f);
     AQIData* aqi = new AQIData[n];
     readData(f,aqi,n);
     std::string s = getenv("QUERY_STRING");
     //e.g. s may be "o = Bubble"
     if(s == "o=Bubble")
     {

             auto t1 = std::chrono::high_resolution_clock::now();
             BubbleSort(aqi,n);
             for (int i = 0; i < n; i++)
                 std::cout << "<html>" << aqi[i].county << " " << aqi[i].AQI<< "<br></html>";
             auto t2 = std::chrono::high_resolution_clock::now();
             auto time_span =
             std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
             std::cout << time_span.count() << "\n";
     }
     else if(s == "o=Selection")
    {

            auto t1 = std::chrono::high_resolution_clock::now();
            SelectionSort(aqi,n);
            for (int i = 0; i < n; i++)
                std::cout << "<html>" << aqi[i].county << " " << aqi[i].AQI<< "<br></html>";
            auto t2 = std::chrono::high_resolution_clock::now();
            auto time_span =
            std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
            std::cout << time_span.count() << "\n";
    }
    else if(s == "o=Insertion")
    {
            auto t1 = std::chrono::high_resolution_clock::now();
            InsertionSort(aqi,n);
            for (int i = 0; i < n; i++)
                std::cout << "<html>" << aqi[i].county << " " << aqi[i].AQI<< "<br></html>";
            auto t2 = std::chrono::high_resolution_clock::now();
            auto time_span =
            std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
            std::cout << time_span.count() << "\n";
    }
    else if(s == "o=QuickSort")
    {
            auto t1 = std::chrono::high_resolution_clock::now();
            qt.Sort(aqi,n);
            for (int i = 0; i < n; i++)
                std::cout << "<html>" << aqi[i].county << " " << aqi[i].AQI<< "<br></html>";
            auto t2 = std::chrono::high_resolution_clock::now();
            auto time_span =
            std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
            std::cout << time_span.count() << "\n";
    }
    else if(s == "o=Merge")
    {
            auto t1 = std::chrono::high_resolution_clock::now();
            ms.Sort(aqi,n);
            for (int i = 0; i < n; i++)
                std::cout << "<html>" << aqi[i].county << " " << aqi[i].AQI<< "<br></html>";
            auto t2 = std::chrono::high_resolution_clock::now();
            auto time_span =
            std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
            std::cout << time_span.count() << "\n";
    }
}
```

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*)

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

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

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

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

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

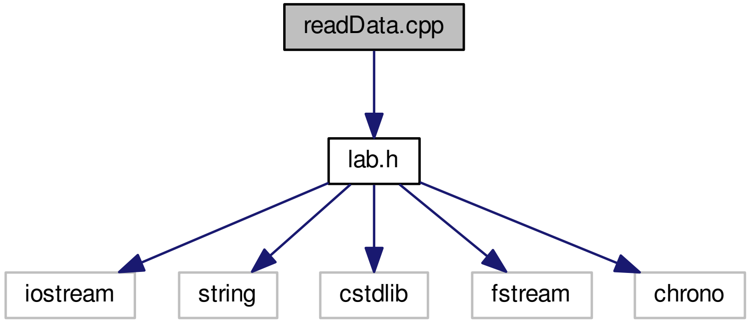
6.5.1.4 bool operator>= (AQIData & *lhs*, AQIData & *rhs*)

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

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*)

```
{

    std::ifstream ifs(f.c_str());
    std::string s; char comma;
    for(int i = 0; i < n; i++)
    {
        getline(ifs,s,',');//reads state
        getline(ifs,aqi[i].county,',');//reads county
        getline(ifs,s,',');//ignores county and reads year
        getline(ifs,s,',');
        getline(ifs,s,',');
        getline(ifs,aqi[i].AQI,',');//ignore year and readsDays

        //ifs >> aqi[i].AQI >> comma;
        getline(ifs,s);
    }
    //aqi[0].county = s;
    ifs.close();


}
```