Lab5

Generated by Doxygen 1.8.8

Sat Nov 4 2017 01:40:59

Contents

1 readData

read some columns from file f snd store them in aqi array

Parameters

f	filename
aqi	array to put data in
n	number of elements in the array
read	the first string in the file f and return it read the file and store the valuse 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

f	The string that is the filename
out	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 chang 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

filename	the filename you want to it to write to
inputstring	the string you want to output

5 readtimes

Void function to read the information in a file and dispaly 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 alogrithms to sort the AQI or air quality index of multiple counties from low to high. Sorting algorithms each have different levels of efficieny 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 alorithm took.

3) UI is very easy and clear to use.

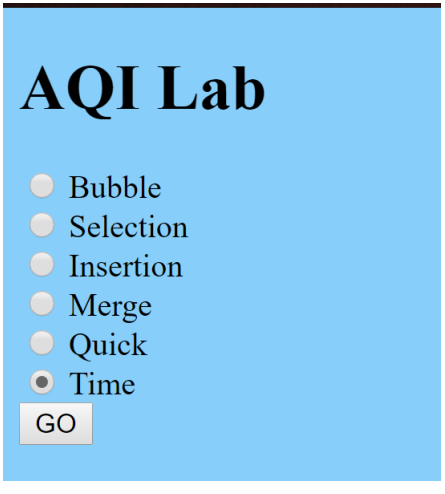
7 Analysis

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. Th 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

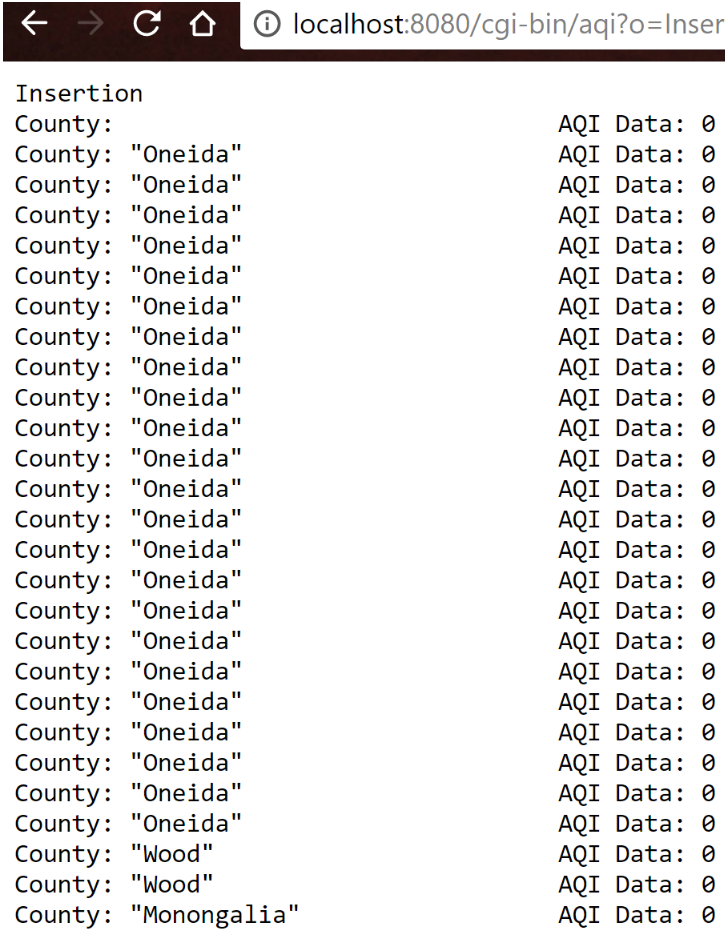
Html was what ws used to create the UI for this lab. we were able to have buttons to select wha we wanted and customize the page very well. Html is widely used in the industry so adopting this platfrom was the propper move. The biggest advantage I see is that I am able to edit, change, and add things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

9 Test



localhost:8080/cgi-bin/aqi?o=Bubbl

```
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
```



```
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: "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
```

localhost:8080/cgi-bin/aqi?o=Quick

```
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
```

localhost:8080/cgi-bin/aqi?o=Selecti..

```
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
```

localhost:8080/cgi-bin/aqi?o=time

```
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:

11 File Index

11.1 File List

Here is a list of all files with brief descriptions:

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]

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

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

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

12.2.2 Member Function Documentation

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

```
{
    int n1, n2;
    AQIData *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];

            Sort(a, n1);
            Sort(a2, n2);
            Merge(a, n1, a2, n2);
        }
}
```

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*)

```
{
    int p;

```

```
    if (n <=2) {
        if (n ==2 && a[1] < a[0])
            Swap(a[0], a[1]);
        }
        else {
            p = Split(a, n);
            Sort(a, p);
            Sort(&a[p+1], n-p-1);
        }
    }
```

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*)

```
{

    ofstream inpfile(filename, ios::app);
    inpfile << inputstring << endl;
    inpfile.close();
    return true;
}
```

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*)

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


     return n;
}
```

13.3 info.cpp File Reference

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

Functions

- int average ()

13.3.1 Function Documentation

13.3.1.1 int average ( )

```
{


}
```

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*)

```
{

    ofstream inpfile(filename, ios::app);
    inpfile << inputstring << endl;
    inpfile.close();
    return true;
}
```

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

```
{
    int i, disorder = n;

    while (disorder) {

        disorder = 0;

        for (i = 1; i < n; i++) {
            if (a[i] < a[i-1]) {

                Swap(a[i], a[i-1]);
                disorder++;

            }
        }
        n--;
    }
}
```

13.4.1.3 int countRows (string *f*)

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


    return n;
}
```

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

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

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

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

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

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

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 chang the authors code

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

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

```
{
```

```
    ifstream ifs (f.c_str());
    string s; char comma = ',';
    getline(ifs,s);
    for( int i = 0; i < n ; i++) {
    getline(ifs,s,','); //read and ignore the state
    getline(ifs, aqi[i].county, ',');
    getline(ifs,s,',');
    getline(ifs,s,','); //read and ignore state
    getline (ifs,s,','); //read amd ind ignore the Days of AQI
    ifs >>  aqi[i].AQI >> comma;
    getline(ifs,s);
}
    ifs.close();
}
```

13.4.1.9 void readtimes ( )

```
{
      string line;
            ifstream inpfile("times.txt");
            if (inpfile.is_open())
            {
                while (getline (inpfile, line) )
                {
                 cout << line << endl;
                 }
             }
        inpfile.close();

}
```

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

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

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 ( )

```
{
    string s = getenv("QUERY_STRING");
    string f = "/home/debian/data/aqi.csv";
    int n = countRows(f);
    AQIData* aqi = new AQIData[n];

    MERGESORT<AQIData> mergesort(n);
    QUICKSORT<AQIData> quicksort;

    readData(f,aqi,n);

    if(s == "o=time")
    {
          readtimes();
    }
    if(s == "o=Bubble")
    {
        cout << "Bubble Sort" << endl;
        auto t1 = std::chrono::high_resolution_clock::now();
        BubbleSort(aqi, n);
        auto t2 = std::chrono::high_resolution_clock::now();
        auto time_span = std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
        double time = time_span.count();
        string str = "Time taken to Bubble sort in seconds: " + std::to_string(time);
        addtimes("times.txt", str);
        for(int i = 0; i < n; i++)
        {
        cout << left << setw(7) << "County: " << left << setw(30) << aqi[i].county;
        cout  <<  "AQI Data: "<< aqi[i].AQI << endl;
```

```
        }
    }
    if(s == "o=Selection")
    {
        cout << "Selection" << endl;
        auto t1 = std::chrono::high_resolution_clock::now();
        SelectionSort(aqi, n);
        auto t2 = std::chrono::high_resolution_clock::now();
        auto time_span = std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
        double time = time_span.count();
        string str = "Time taken to Selection sort in seconds: " + std::to_string(time);
        addtimes("times.txt", str);
        for(int i = 0; i < n; i++)
        {
        cout << left << setw(7) << "County: " << left << setw(30) << aqi[i].
    county;
        cout  <<  "AQI Data: "<< aqi[i].AQI << endl;
        }
    }
    if(s == "o=Insertion")
    {
        cout << "Insertion" << endl;
        auto t1 = std::chrono::high_resolution_clock::now();
        InsertionSort(aqi, n);
        auto t2 = std::chrono::high_resolution_clock::now();
        auto time_span = std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
        double time = time_span.count();
        string str = "Time taken to Insertion sort in seconds: " + std::to_string(time);
        addtimes("times.txt", str);
        for(int i = 0; i < n; i++)
        {
        cout << left << setw(7) << "County: " << left << setw(30) << aqi[i].
```

```
county;
        cout  <<  "AQI Data: "<< aqi[i].AQI << endl;
        }
    }
    if(s == "o=Merge")
    {
        cout << "Merge Sort" << endl;
        auto t1 = std::chrono::high_resolution_clock::now();
        mergesort.Sort(aqi, n);
        auto t2 = std::chrono::high_resolution_clock::now();
        auto time_span = std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
        double time = time_span.count();
        string str = "Time taken to Merge sort in seconds: " + std::to_string(time);
        addtimes("times.txt", str);
        for(int i = 0; i < n; i++)
        {
        cout << left << setw(7) << "County: " << left << setw(30) << aqi[i].
county;
        cout  <<  "AQI Data: "<< aqi[i].AQI << endl;
        }
       }
        if(s == "o=Quick")
    {
        cout << "Quick Sort" << endl;
        auto t1 = std::chrono::high_resolution_clock::now();
         quicksort.Sort(aqi, n);
         auto t2 = std::chrono::high_resolution_clock::now();
         auto time_span = std::chrono::duration_cast<std::chrono::duration<double>>(t2-t1);
         double time = time_span.count();
         string str = "Time taken to Quick sort in seconds: " + std::to_string(time);
         addtimes("times.txt", str);
         for(int i = 0; i < n; i++)
```

```
        {
        cout << left << setw(7) << "County: " << left << setw(30) << aqi[i].county;
        cout  <<  "AQI Data: "<< aqi[i].AQI << endl;
        }
    }

    return 0;
}
```

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*)

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

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

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

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

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

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 chang the authors code

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

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*)

```
{
    ifstream ifs (f.c_str());
    string s; char comma = ',';
    getline(ifs,s);
    for( int i = 0; i < n ; i++) {
    getline(ifs,s,','); //read and ignore the state
    getline(ifs, aqi[i].county, ',');
    getline(ifs,s,',');
    getline(ifs,s,','); //read and ignore state
    getline (ifs,s,','); //read amd ind ignore the Days of AQI
    ifs >>  aqi[i].AQI >> comma;
    getline(ifs,s);
}
    ifs.close();
}
```

13.8 readtimes.cpp File Reference

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

Functions

- void readtimes ()

13.8.1 Function Documentation

13.8.1.1 void readtimes ( )

```
{
      string line;
            ifstream inpfile("times.txt");
            if (inpfile.is_open())
            {
                while (getline (inpfile, line) )
                {
                 cout << line << endl;
                 }
             }
        inpfile.close();

}
```

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*)

```
{
    int i, disorder = n;

    while (disorder) {

        disorder = 0;

        for (i = 1; i < n; i++) {
            if (a[i] < a[i-1]) {

                Swap(a[i], a[i-1]);
                disorder++;

            }
        }
        n--;
    }
}
```

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

```
{

    int i, j;
    AQIData aCurrent;

    for (i =1; i < n; i++) {

```

```
        //Save the current element:
        aCurrent = a[i];

        //Find location j where it should be inserted
        //among the first i-1 elements:
        for (j=0; j < i; j++)
            if (a[j] >= aCurrent) break;

        for (int k = i-1; k >= j; k--)
            a[k+1] = a[k];

        //Insert saved element where it belongs:
        a[j] = aCurrent;
    }
}
```

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

```
{
    int i, iMax;

    while ( n > 1) {
        //Find the largest element:
    for (iMax = 0, i = 1; i < n; i++)
        if (a[i] > a[iMax]) iMax = i;

    //Swap it with the last element:
    Swap(a[iMax], a[n-1]);
    n--;
}
}
```

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

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

13.10 specification.dox File Reference