C++ Coursework 1

Dequn Teng 201448415

Specification Demonstration

Input

There are a number of files, containing an unknown number of temperatures at different time, and one possible file has the sample data format as follows. For example, there are two possible input files, as follows.

Time	Temperature
0	25
0.1	26
0.2	28
0.3	30
0.4	35

Time	Temperature
0	17
0.1	25
0.2	14
0.3	31
0.4	14
0.5	36
0.6	38.4
0.7	40.8
0.8	43.2
0.9	45.6

There are a few points to be noticed here.

- The temperature does not necessarily have to be integers.
- The temperature does not necessarily go increasing as shown in the data table in the task sheet
- The number of data point is not known in advance.
- The time interval is 0.1s

Processing Logic

The programme must calculate the average temperature from all the data read from the files at a given time, and we construct the corresponding intermediate node which is as follows.

	0 file	1	2	3	4

	0 file	1	2	3	4
0 time	32	52	72	92	112
1	26	24	22	20	18
2	52	62	72	82	92
3	34	15			
4	62	32			

Starting with the initial table

	0	1	2	3	4
0					

vector<vector> finalTest;

Then let's talk about the corresponding value to be added to the table, the corresponding data will be added with the format as follows.

	0	1	2	3	4
0	1	1	1	2	2
1	2	3	-	-	-
2	2	-	-	-	-

Which is achieved by the c++ vector characteristics and especially the 2D vector.

Output

The output file should have three columns, namely the time, the averaged temperature, and number of data points used in that average. For example, if 3 out of 5 files have data points at 1.5 s, the third column for t = 1.5 s should have the number 3, while for t = 0 s the third column should have 5.

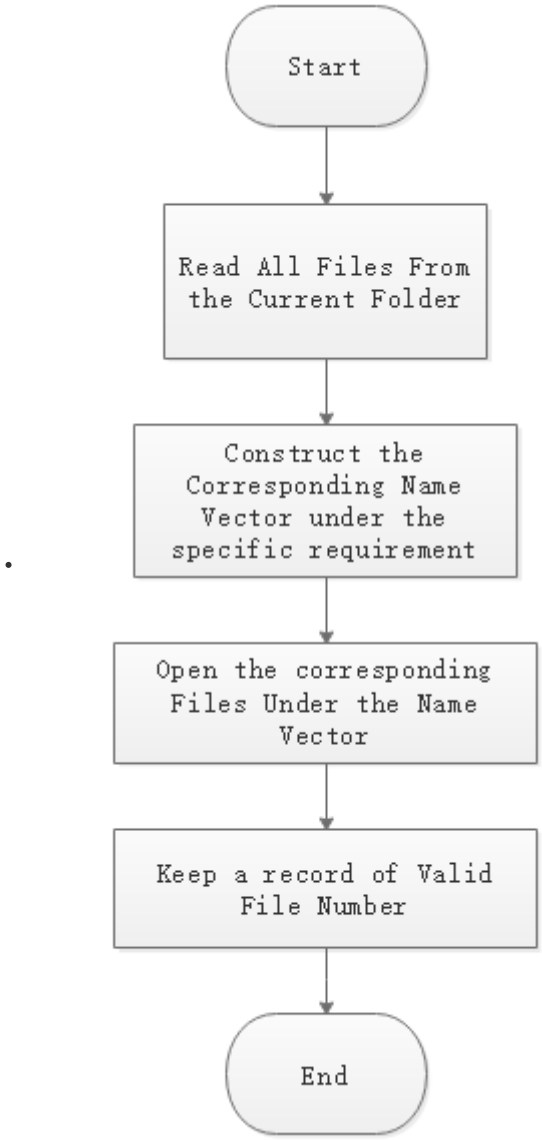
The corresponding output of this file is

Time	Average Temperature	Num of Data Points
0	12.5	2
0.1	13.05	2
0.2	14.1	2
0.3	15.15	2
0.4	17.7	2
0.5	36	1
0.6	38.4	1
0.7	40.8	1
0.8	43.2	1
0.9	45.6	1

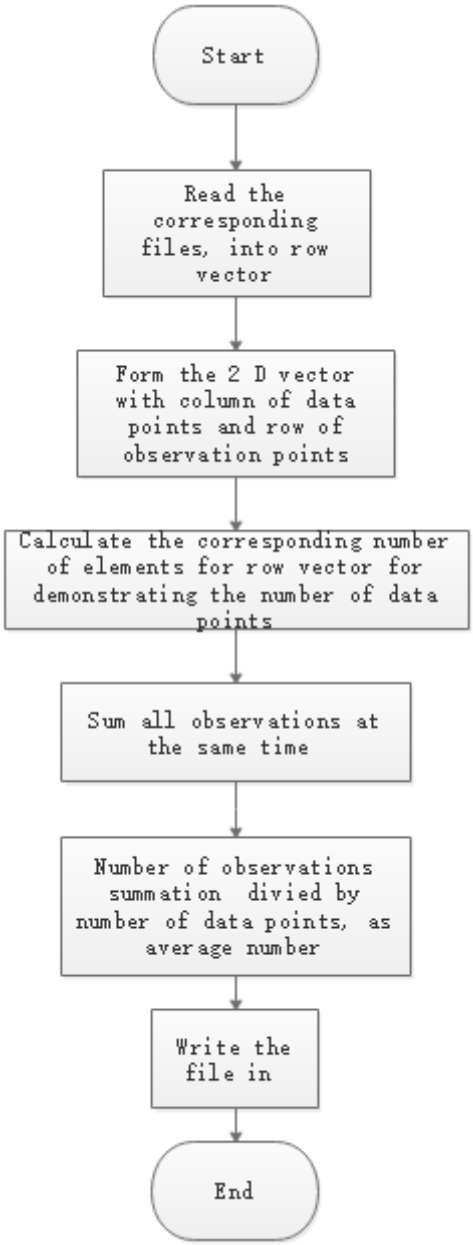
Flowchart

- Read every data from the corresponding file to vectors, and the number of data in the file determines the number of data in the vector
- Get the size of the each vectors, and store in the corresponding size vector, which may be used to determine the corresponding number of data used in this program
- If there is a number in this vector, the program should convert the number to 1, which is used to determine the number of data digit used in this calculation, which is called the numVector in this case.
- Up to now, we are known of the following.
    - Each vector representing the values within the file, which are named as the corresponding name, called, Experiment1[i], Experiment2[i], ...respectively.
    - The corresponding size of the vector
    - The corresponding binary vector, representing whether there is value at the corresponding position numVector1[i], numVector2[i], ...respectively.
- The corresponding output of the program is demonstrated as follows.
    - Time should be a vector, increased by a step size of 0.1, up to most number contained in one vector, represented by the variable largestNumberInAll, in this case, the largest time should be 0.1*largestNumberInAll
    - The averaged temperature vector called averagedTemperature[i] should be calculated by a for loop
        - for each variable i
        - calculate the corresponding averagedTemperature[i]=experiment

Preprocessing Flowchart



Logic Processing Unit



User Instructions

- The user should click the .exe file for demonstrating and the program will automatically read in all data under the current folder named by the consecutive naming scheme "Experiment1.csv", "Experiment2.csv"... And it will generate the "output.csv" file named to demonstrate the corresponding user
- If the user uses the IDE like the visual studio for debugging purposes, the console will generate the corresponding

Testing and Verification Results

Unit File Testing

- Input

-

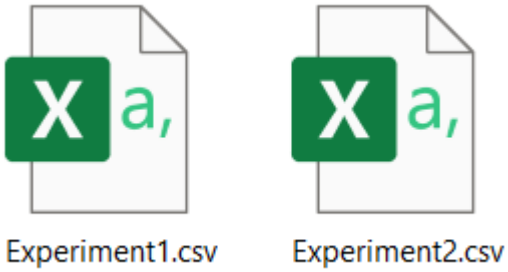
Time	Temperature
0	71
0.1	50
0.2	59

- Output
-

Recorded	Averaged	Measured	DataPoints
0	71	1	
0.1	50	1	
0.2	59	1	

- Explanation
- The input time and temperature is matching with the data points recorded as the 1

Double Files Testing

- Input: Experiment2
- 

-

	Time	Temperature
2	0	87
3	0.1	42
4	0.2	67
5		

-

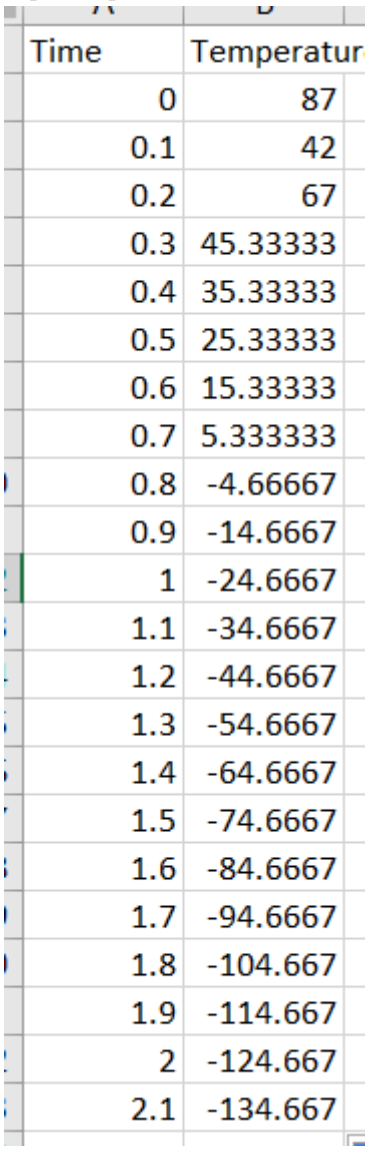
Time	Temperature
0	71
0.1	50
0.2	59

- Output
-

Recorded	Averaged	Measured	DataPoints
0	79	2	
0.1	46	2	
0.2	63	2	

- Explanation
- The output is matching the measured case from the input file

Three Files and Decimal Testing

- Input: Experiment3
- 

Time	Temperatur
0	87
0.1	42
0.2	67
0.3	45.33333
0.4	35.33333
0.5	25.33333
0.6	15.33333
0.7	5.333333
0.8	-4.66667
0.9	-14.6667
1	-24.6667
1.1	-34.6667
1.2	-44.6667
1.3	-54.6667
1.4	-64.6667
1.5	-74.6667
1.6	-84.6667
1.7	-94.6667
1.8	-104.667
1.9	-114.667
2	-124.667
2.1	-134.667

- Output

Recorded	Averaged	Measured DataPoints
0	81.66667	3
0.1	44.66667	3
0.2	64.33333	3
0.3	45	1
0.4	35	1
0.5	25	1
0.6	15	1
0.7	5	1
0.8	-4	1
0.9	-14	1
1	-24	1
1.1	-34	1
1.2	-44	1
1.3	-54	1
1.4	-64	1
1.5	-74	1
1.6	-84	1
1.7	-94	1
1.8	-104	1
1.9	-114	1
2	-124	1
2.1	-134	1

- Explanation
- This function works well

Extreme Case Testing

- Input: Experiment4
- Input

102	10	74
103	10.1	76
104	10.2	34
105	10.3	70
106	10.4	52
107	10.5	78
108	10.6	78
109	10.7	89
110	10.8	63
111	10.9	77

- Output

98	9.6	16	1
99	9.7	76	1
100	9.8	62	1
101	9.9	53	1
102	10	74	1
103	10.1	76	1
104	10.2	34	1
105	10.3	70	1
106	10.4	52	1
107	10.5	78	1
108	10.6	78	1
109	10.7	89	1
110	10.8	63	1
111	10.9	77	1

- Explanation
- Pass the test

Appendix: Source Code

```
```
// Library Declaration

#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <numeric>
#include "stdio.h"

// Name Space Declaration

using namespace std;

//Variable Declaration
string fileName;
vector <int> tempVec;
vector<vector<int> > stored2DVector;
int numOfFiles;
int keepTemp(string data);
void readFile(int fileNumber);
int fileCount();
int findMaxRow(int fileNumber);
void calculationModule(int input);

//Main Function with the supporting functions
```
```

```
int main()
{
    fileCount(); // count number of file
    readFile(numOfFiles); // read the corresponding files
    int maxLineNumber = findMaxRow(numOfFiles) - 1; //Get the corresponding max
line number to determine the time
    calculationModule(maxLineNumber); // calculate the corresponding data point and
average point
    system("pause");// Hold the console for the user interface
};



//return the max number of rows contained in the file, which is used for
determining the combined time

int findMaxRow(int count)
{
    //Variable Declarization
    vector<int> numberOfRows;
    string b;
    //File Read
    for (int i = 0; i < count; i++)
    {
        //Start with 1
        int temp = i + 1;
        //String combination
        b = "Experiment" + to_string(temp) + ".csv";
        //variable initization
        int rows = 0;
        ifstream file(b);
        string line;
        //read the corresponding rows
        while (getline(file, line))
            rows++;
        //push back corresponding row to the number of rows vector
        numberOfRows.push_back(rows);
    }
    //Max Value Determination
    int max = 0;
    //return the max value
    for (int i = 0; i < count; i++)
    {
        if (max < numberOfRows[i])
        {
            max = numberOfRows[i];
        }
    }
    return max;
}

//Find the corresponding files contained within the file

int fileCount()
```

```
{
    //for loop
    for (int i = 1;; i++)
    {
        //generate the corresponding fileName
        fileName = "Experiment" + to_string(i) + ".csv";
        //if the file is open
        ifstream myfile(fileName);
        if (myfile.is_open())
        {
            //increase the number of files by one
            numOfFiles++;
            //close myfile
            myfile.close();
        }
        else
        {
            //close myfile
            myfile.close();
            return numOfFiles;
        }
    }
    //return the number of files
    return numOfFiles;
}

// read file function
void readFile(int fileNumber)
{

    for (int i = 0; i <= fileNumber; i++)
    {
        //variable initizalization
        string headLine;
        string data;
        ifstream myfile(fileName);
        //generate the corresponding file name
        fileName = "Experiment" + to_string(i + 1) + ".csv";
        //read the headline of the file
        getline(myfile, headLine);
        //if there myfile is not empty
        while (myfile)
        {
            //continuing reading the file
            getline(myfile, data);
            //keep only the valided data
            int temp = keepTemp(data);
            //push the valid data to the vector
            tempVec.push_back(temp);
        }
        //push the vector back
        stored2DVector.push_back(tempVec);
        //clear the vector
        tempVec.clear();
    }
```

```
    //initialization
    for (int i = 1; i <= fileNumber; i++)
    {
        //set the column
        int column = stored2DVector[i].capacity();
        //initialize the first row and column
        stored2DVector[i][column - 1] = 0;
    }

}

// Function to only only extract the temperature
int keepTemp(string input)
{
    //examine the input
    while (input != "")
    {
        //find delimiter
        int delimiter = input.find(",");
        //find the pure temperature
        string leftString = input.erase(0, delimiter + 1);

        //cout << leftString << "!!!!!!!!!!!!!";

        int output = stod(leftString);
        //return the corresponding output
        return output;
    }
}

void calculationModule(int input)
{
    //Generatet the corresponding output file
    ofstream newfile("output.csv");
    //Set the corresponding column name
    newfile << "Recorded Times,Averaged Temperature,Measured DataPoints" << '\n';
    //store it to the two dimension vector
    for (int i = 0; i < input; i++)
    {
        //variable ini
        double mean;
        int summation = 0;
        double measuredPoint = 0;
        //store it to the one dimension vector
        for (int j = 1; j <= numOfFiles; j++)
        {
            if (i + 1 < stored2DVector[j].size())
            {
                //Add the corresponding summation
                summation += stored2DVector[j][i];
                //increase the measued data point by one
                measuredPoint += 1;
            }
        }
        //calcultating of the mean
```

```
        mean = summation / measuredPoint;
        //output the information to the file
        newfile << 0.1 * i << "," << to_string(mean) << "," <<
to_string(measuredPoint) << '\n';
        //print to the console for the debug purposes
        cout << mean << "|" << measuredPoint << '\n';
    }
}
```