**Final Project**

**Katherine Sandoval**

**MAC 101**

**Introduction**

In this interesting and challenging project there are three basic objectives that coincide with determining the averages of people and quizzes of a class. The first objective of this project is to calculate the total average for each of the students’ quizzes as well as the quizzes’ total average. The second objective of this project consists of dropping the highest and the lowest scores for both the quizzes and the students in order to calculate the total average of the reminding scores. The final objective of this project is to calculate the grand averages which are the total averages for all the averages already found. Everything in this project is based on the information given of a class’ test scores. In order to achieve the objectives of this project, it is necessary to have some basic knowledge of the important concepts of strings, functions, while and for loops, two dimensional arrays, converter function from string to integer, and insertion sort function.

**Methodology**

In order to do this project, there has to be five libraries included; they are iostream, fstream, string, sstream and iomanip.

The first step is to create two global variables and define their sizes.  They hold both, the total number of students (20 rows) and the total number of quizzes (8 columns).

The second step is to create a two dimensional global array in which all the data is held.

After declaring all the variables, the prototypes for all functions that will be giving the results have to be declared.

In the main function, the first step is to import the file.  In order to do so, it is necessary to use an "ifstream"; then to declare a string which would be holding the information that is being input as a form of string.

After inputting the file, it is read in the program as a string with a nested while loop.  The first while loop takes each of the lines and the second while loop takes each of the numbers and both place the values into the variables rows and columns already declared.

Before placing the values into the two dimensional array, it is necessary to convert them from string to integers in order to realize all operations.  To do so, the "atoi" function has to be used. After the program recognizes them as integers, they have to be placed into the two dimensional array to perform all the operations necessary.

After the nested while loops, the main function has to call the printArray function which gives all the results.

From now on, it is a matter of creating functions in order to calculate all the averages.

The first function called "avgCol" calculates the average for every quiz (straight column average). The return type of this function is double since it gives back a decimal number. This function goes through every row of the column number that is being placed on the constant c from the printArray and returns that average.

The second function called "grandAvgCol" calculates the average of all the quizzes' averages (grand average). The return type of this function is double since it returns a decimal number.  This function gets all the averages by calling the "avgCol" function. They are placed and sum in a variable called grandAvg which at the end is divided by the number of columns. The grand average is finally returned.

The third function called "avgRows" calculates the average for every student (straight row average). The return type of this function is double since it gives back a decimal number. This function goes through every column of the row number that is being place on the constant r from the printArray and returns that average.

The fourth function called "grandAvgRows" calculates the average of all the students' averages (grand average). The return type of this function is double since it returns a decimal number.  This function gets all the averages by calling the "avgRows" function. They are placed and sum in a variable called grandAvg which at the end is divided by the number of rows.  The grand average is finally returned.

The fifth function called "insertionSort" is a simple sorting algorithm that builds the final sorted array one item at a time.  Insertion sort iterates, consuming one input element each repetition, and growing a sorted output list. On a repetition, insertion sort removes one element from the input data, finds the location it belongs within the sorted list, and inserts it there. It repeats until no input elements remain. Sorting is typically done in-place, by iterating up the array, growing the sorted list behind it. At each array-position, it checks the value there against the largest value in the sorted list (which happens to be next to it, in the previous array-position checked). If larger, it leaves the element in place and moves to the next. If smaller, it finds the correct position within the sorted list, shifts all the larger values up to make a space, and inserts into that correct position. The resulting array after k iterations has the property where the first k + 1 entries are sorted ("+1" because the first entry is skipped). In each iteration the first remaining entry of the input is removed, and inserted into the result at the correct position.

The sixth function called "studentAvgDrop" excludes the lowest and the highest scores of each student and finds the average of the remaining grades.  This functions uses the insertionSort function to sort each row and to be able to start the for loop on the element 1 and end on column -1 to drop the highest and the lowest grades. This happens when going through every column of the row number that is being place on the constant r from the printArray. It finally returns the average.

The seventh function called "grandAvgDropRows" calculates the average of all students’ (rows) averages after dropping the highest and the lowest scores. The return type of this function is double since it returns a decimal number.  This function gets all the averages by calling the "studentAvgDrop" function. They are placed and sum in a variable called granAvg which is divided by the number of rows.  And finally, this function returns the grand average.

The eighth function called "quizAvgDrop" excludes the lowest and the highest scores of each quiz and finds the average of the remaining grades.  This functions uses the insertionSort function to sort each column and to be able to start the for loop on the element 1 and end on row -1 to drop the highest and the lowest grades. This happens when going through every row of the column number that is being place on the constant c from the printArray. It finally returns the average.

The ninth function called "grandAvgDropCols" calculates the average of all quizzes' (columns) averages after dropping the highest and the lowest scores. The return type of this function is double since it returns a decimal number.  This function gets all the averages by calling the "quizAvgDrop" function. They are placed and sum in a variable called granAvg which is divided by the number of columns. And finally, this function returns the grand average.

The tenth and last function called "printArray" prints out all the elements of the two dimensional array using a nested for loop and then calls every function using for loops and prints out all final results. Since this function doesn't return a value, it has to be declared as a void.

**Conclusion**

In the achievement of the three objectives necessary for the success of this project, it required the use of vital concepts which helps to form the foundation for the development of this project. These concepts consist of strings when getting the data from a csv file, and a converter function called “atoi” which converts the elements from strings to integers in order to place them as integers into the two dimensional array. In order to perform all calculations for the three objectives it is necessary the concept of functions which access the two dimensional array using while and for loops and process the calculations. For the second objective it is necessary the insertion sort function in order to sort the array and finally drop the highest and lowest grades. For the third objective it is necessary to know how to call a function and place its result into the function to calculate the grand averages. When each concept is organized in the right place by following a sequence of logic, then the expected output is achieved. For the first objective which was to find the averages of the quizzes and students, most of the averages for each student seemed to vary from zero to six. In the same way, the averages for each of the quizzes seemed to vary from one to six. The second objective that consisted in dropping the highest and lowest grades for each the students and the quizzes both seemed to vary from zero to seven, not having much of a difference when dropping or not certain grades. For the third objective in which the grand averages had to be calculated, all four grand averages were very similar among them by giving a result from 4.31 to 4.4. All the objectives for this project were successfully achieved.

**Appendix**

#include <iostream>

#include <fstream>

#include <string>

#include <sstream>

#include <iomanip>

using namespace std;

#define rows 20 // define a constant for students

#define columns 8 // define a constant for quizzes

int grades[rows][columns]; // declare a global array

// Prototypes

double avgCol(int scores[][columns], int c);

double grandAvgCol (int scores[][columns]);

double avgRows(int scores[][columns], int r);

double grandAvgRows(int scores[][columns]);

int \*insertionSort(int arr[], int length);

double studentAvgDrop(int scores[][columns], int r);

double grandAvgDropRows(int scores[][columns]);

double quizAvgDrop(int scores[][columns], int c);

double grandAvgDropCols(int scores[][columns]);

void printArray(int scores[][columns]);

int main ()

{

ifstream my\_file("E:\\grades.csv"); // get the input file

string line; // place holder for the string; hold the rows

int row = 0; // set rows to zero

int column = 0; // set columns to zero

while (getline(my\_file, line)) // this while loop takes the source file; this

one gives a line

{

stringstream linestream(line); // convert

string item\_string; // convert

column = 0; // set column to zero each time to loop again

while (getline(linestream, item\_string, ',')) // this while loop gives a

number

{

int number = atoi(item\_string.c\_str()); // atoi function converts the array

from a string to an integer

grades[row][column] = number; // place value number into the array

column++; // left to right top to bottom

}

row++;

}

printArray(grades); // final print

system ("pause");

return 0;

}

/\*

This function calculates the average for every quiz (straight column average)

Precondition: all the elements of the array have been filled

Post condition: the average of these elements is computed and returned

\*/

double avgCol(int scores[][columns], int c)

{

// Declare variables

int j;

double avg;

double sum = 0; // initialize sum

for (j = 0; j < rows; j++) // read every row in each column

{

sum += grades[j][c];

avg = (double) sum / rows; // calculate average for each quiz

}

return avg;

}

/\*

This function calculates the grand average of all quizzes (columns)

(The average of all the averages)

Precondition: all the averages of avgCol are computed and pass to this function

Post condition: the average of these elements is computed and returned

\*/

double grandAvgCol (int scores[][columns])

{

double grandAvg = 0; // initialize grandAvg

for (int i = 0; i < columns; i++) // read every column

grandAvg += avgCol(scores, i); // call avgCol function and place the

result in grandAvg

grandAvg = grandAvg / (double)columns; // calculate grandAvg

return grandAvg;

}

/\*

This function calculates the average for every student (straight row average)

Precondition: all the elements of the array have been filled

Post condition: the average of these elements is computed and returned

\*/

double avgRows(int scores[][columns], int r)

{

// declare variables

int j;

double avg;

double sum = 0; // initialize sum

for (j = 0; j < columns; j++) // read every column in each row

{

sum += scores[r][j];

avg = (double)sum / columns; // calculate average for each student

}

return avg;

}

/\*

This function calculates the grand average of all students (rows)

(The average of all the averages)

Precondition: all the averages of avgRows are computed and pass to this function

Post condition: the average of these elements is computed and returned

\*/

double grandAvgRows (int scores[][columns])

{

double grandAvg = 0; // initialize grandAvg

for(int i = 0; i < rows; i++) // read every row

grandAvg += avgRows(scores, i); // call avgRows function and place the

result in grandAvg

grandAvg = grandAvg / (double)rows; // calculate grandAvg

return grandAvg;

}

/\*

Insertion sort is a simple sorting algorithm that builds the final sorted

array (or list) one item at a time.

Precondition: all the elements of the array have been filled

Post condition: this iteration will insert grades[rows][columns] into the

sorted order

\*/

int \*insertionSort(int arr[], int length)

{

int i, j, tmp;

for (i = 1; i < length; i++)

{

for(j = i; j > 0 && arr[j - 1] > arr[j]; j--)

{

tmp = arr[j];

arr[j] = arr[j - 1];

arr[j - 1] = tmp;

}

}

return arr;

}

/\*

This function excludes the lowest and highest scores for each student and finds

the average of the remaining grades.

This function will use the insertionSort function as a pointer.

Precondition: all the elements of the array have been filled; elements must be

Sorted.

Post condition: the average of these elements is computed and returned.

\*/

double studentAvgDrop(int scores[][columns], int r)

{

double sum = 0;

double avg;

int \*tmp = new int[columns]; // initializing tmp of size column in memory

for (int j = 0; j < columns; j++)

{

tmp[j] = scores[r][j]; // Insert row into temp array

}

tmp = insertionSort(tmp, columns); // Sort array

sum = 0;

for (int j = 1; j < columns - 1; j++) // Find avg by skipping the first

and last element

{

sum += tmp[j];

}

avg = (double)sum / (columns - 2); // calculate the average

delete tmp;

return avg;

}

/\*

This function calculates the grand average of all the students (rows)

after dropping the highest and lowest scores (the average of all the averages).

Precondition: all the averages of studentAvgDrop are computed and pass to this

function.

Post condition: the average of these elements is computed and returned

\*/

double grandAvgDropRows(int scores[][columns])

{

int i;

double grandAvg = 0; // initialize grandAvg

for(i = 0; i < rows; i++) // read every row

grandAvg += studentAvgDrop(scores, i); // call studentAvgDrop function

and place the result in grandAvg

grandAvg = grandAvg / (double)rows; // calculate grandAvg

return grandAvg;

}

/\*

This function excludes the lowest and highest scores for each quiz and finds the

average of the remaining grades.

This function will use the insertionSort function as a pointer.

Precondition: all the elements of the array have been filled; elements must be

sorted.

Post condition: the average of these elements is computed and returned

\*/

double quizAvgDrop(int scores[][columns], int c)

{

double avg;

double sum = 0;

int \*tmp = new int[rows]; // initializing tmp of size rows in memory

for (int j = 0; j < rows; j++)

{

tmp[j] = scores[j][c]; // Insert row into temp array

}

tmp = insertionSort(tmp, rows); // Sort array

sum = 0;

for (int i = 1; i < rows - 1; i++) // Find avg by skipping the first and

last element

{

sum += tmp[i];

}

avg = (double)sum / (rows - 2); // calculate the average

delete tmp;

return avg;

}

/\*

This function calculates the grand average of all the quizzes (columns) after

dropping the highest and lowest scores. (The average of all the averages)

Precondition: all the averages of quizAvgDrop are computed and pass to this

function.

Post condition: the average of these elements is computed and returned

\*/

double grandAvgDropCols(int scores[][columns])

{

int i;

double grandAvg = 0; // initialize grandAvg

for(i = 0; i < columns; i++) // read every column

grandAvg += quizAvgDrop(scores, i); // call quizAvgDrop function and place

the result in grandAvg

grandAvg = grandAvg / (double)columns; // calculate grandAvg

return grandAvg;

}

/\*

This function prints out all numbers in the array as well as all the computed

averages. (it is final print; prints all calculations)

Precondition: all the elements of the array have been filled.

All functions have to be called

Post Condition: all averages and calculations are printed

\*/

void printArray(int scores[][columns])

{

int i, j;

for(i = 0; i < rows; i++)

{

for (j = 0; j < columns; j++)

cout << scores[i][j] << "\t";

cout << setprecision(2) << fixed << avgRows(grades, i) << "\t";

cout << setprecision(2) << fixed << studentAvgDrop(grades, i) << endl;

}

for(int i = 0; i < columns; i++)

cout << "\t";

cout << setprecision(2) << fixed << grandAvgRows(grades) << "\t";

cout << setprecision(2) << fixed << grandAvgDropRows(grades) << endl << endl;

for(int j = 0; j < columns; j++)

cout << avgCol (grades, j) << "\t";

cout << setprecision(2) << fixed << grandAvgCol(grades) << endl;

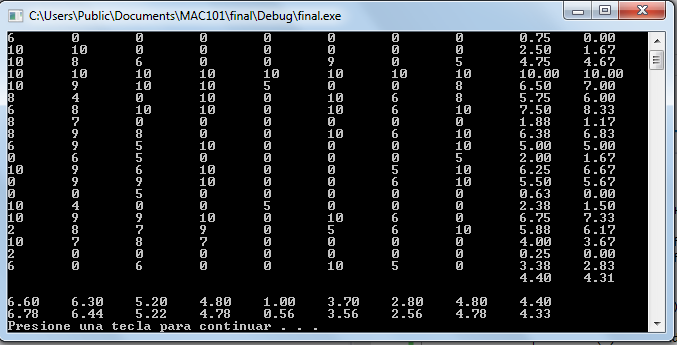
for(int j = 0; j < columns; j++)

cout << quizAvgDrop(grades, j) << "\t";

cout << setprecision(2) << fixed << grandAvgDropCols(grades) << endl;

}

**Result**

****

**References**

1. Alex. “Multidimensional Arrays” LearnCpp.com. July 6, 2007.
2. Algorithms and Data Structures with implementations in java and c++. [www.algolist.net](http://www.algolist.net).
3. Cplusplus.com
4. Jacobson, Lex and John Even, “ Multi-Dimensional Arrays.” July 24, 1996.
5. Tutorialspoint.com.
6. Overland, Brian. “C++ Without Fear”. Second Edition. April 2011.