1. **MAIN**

In the main function we have our declaration of all the 2-dimensional arrays that we require for the task. Followed by some variables that need to be used by several of the functions.

Next is the menu and switch case loop. Inside the loop a function is called that shows the menu options to the user and then return the choice of the user. This choice is passed into a switch case which then goes on to call the appropriate functions that the user requires.

At the end of the main we delete the newly created 2-dimensional arrays so as to avoid memory leaking.

Text

Description automatically generated

1. **mainMenuOptions( )**

This is a simple function that displays the options the user has. Once the user inputs an option, the user input is saved and passed on to a variable in the main.

Text

Description automatically generated

1. **readIn( )**

The readIn function takes in almost all the variables declared in main. It takes it the address of the variable so that any modifications to the variables will be reflected back into the main and this will allow us to pass the same values into other functions. The readIn function’s job is to extract all the information in the configuration file and split it up accordingly. Through fstreams, vectors and the splitString function, the data is extracted into variables and the variables are passed on to the next function buildCityMap.

Text

Description automatically generated

Text

Description automatically generated

1. **buildCityMap( )**

This is the function that builds the 2dimensional arrays from the given parameters. Eg. Maximum x-axis, maximum y-axis…etc.

First all the 2D arrays are initialized through dynamic memory allocation so that the array dimensions are dynamic and not static. For loops are then used to build the hex border, and then add the x and y coordinates. This function builds the 2d array skeleton for all the 2-dimensional arrays declared in the main. Once all the arrays are initialized and built, it is then passed onto the next functions. They are the addCityLoc, buildPressureMap and buildCloudMap

Text

Description automatically generated

Graphical user interface, text, chat or text message

Description automatically generated

1. **addCityLocs( )**

This function receives the citylocation file and fills the city map with the different cities and the corresponding coordinates. Fstream and vectors are used again to extract the data from the cityLocation file. Some calculations were required when plotting the grids as the coordinates were required to be offset so as to be at the correct place of the grid. Once the offset was calculated it can be applied to all the coordinates in the cityLoc file so as to plot the true coordinates.

Text

Description automatically generated

1. **buildCloudMap( )**

This function is similar to addCityLocs. However in this function 3 different grids are being built. Cloud map with the absolute value given in the file, Cloud map with the index and also cloud map with the LMH values. The principle follows the addCityLocs function, the coordinates are extracted and then offsetted to the true coordinates then the value is added into the appropriate 2-dimensional array. Instead of using multiple if-else statements for the cloud index calculation, I realized I could just extract the digit in the tens position of the cloud cover value, so this saved some time.

Text

Description automatically generated

1. buildPressureMap( )

Exactly the same as buildCloudMap, just extracting data from a different file.

Text

Description automatically generated

1. summaryReport( )

The summary report collates all the information extracted from the files. I felt it was easier to do all the calculations at the last step so that I would not need to pass so much information through the functions. This function begins with extracting only the cityID and cityName from the cityLocation file. I then use the sort and unique function to only get distinct values. It is also here that I create the parameter for each city. I do this through a series of if statements, checking each grid surrounding the city. Once the perimeter is constructed for the city, I cross check it with the absolute cloud and absolute pressure grids so as to calculate my avg pressure and avg clouds. Once this is done for the city, I remove the perimeter and do the same for the next city. The rest of the function is displaying and formatting the information.

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

1. displayCityMap( )

Simple function that takes in the grid as a parameter and outputs it as a 2-dimensional array for viewing.

Text

Description automatically generated

**Summary of what went wrong**

In the old implementation of the code the major problem was that the way I built the 2D arrays for the city grid. Instead of building it as [YAxis][XAxis] I built it as [XAxis][YAxis] even though the rest of my code added values into my 2D array as [Y][X]. this caused my x-axis to be the size of y, and my y-Axis to be the size of x, causing multiple segmentation errors. Once that problem was solved, and the arrays were initialized correctly, the code was able to run smoothly.

Old New

Text

Description automatically generatedText

Description automatically generated

Another problem I found was that my code did not cater for 2 digit numbers in the coordinates, so some lines of code were substituted for better ones.

old

Text

Description automatically generated

new

Text

Description automatically generated

Last but not least, I added a few lines of code to ignore comments and to ignore empty lines not only in the configuration file but also in all the other files.

Text

Description automatically generated