Rationale

In order to successfully finish the project we would have to:

- Read multiple weather data files line by line.

- Eliminate duplicate registers.

- Make a sensible use of a Binary Search Tree and a Map as data structures.

- Calculate max wind speed for year and month

- Calculate Average wind speed for year

- Calculate Total solar radiation for year

- Save into a file the Average wind speed + solar radiation for year

- Calculate Max solar radiation in a day

- Be able to exit program

- Don't add registers that have value zero to the calculation of average wind speed for a year.

- Don't add registers that have value lesser that 100 to the calculation of Solar Radiation for an year.

- Display all times that certain max solar radiation occurred in a given day.

So, the following is the data structures approach that I followed for each of those requirements

Read multiple weather data files line by line.
First I Create a class called Metadata that works like a log object. So, all registers including date, time, solar radiation and wind speed are

collected from a single line and set as the parameters of a Metadata object.
At the end of reading the file, when all parameters were set, I had this object and I had to choose a data structure to store them.
So, I saved the object first in a Vector which includes all registers collected, in other words, all metadata objects that I created, this Vector them will be used for the calculations of the functions one to four from the menu displayed to the user.

Eliminate duplicate registers

The way a figure this out, was to while I was inserting the data to the Vector, while reading the files, I also would insert the metadata object to a binary tree of Metadata objects, so, because I had overload the operators < > == != at the Metadata class, so this tree would only keep unique registers, then I created a token which I called tokenDuplo, then I used the method search, from my BST, next time that it iterates it would check if the metadata object, which was made unique by comparing date and time respectively, if it was already present in the BST, then if not the Token gets the value of one and is used to avoid insertions in the vector.

Make a sensible use of a Binary Search Tree and a Map as data structures

These 2 data structures came very handy to allow me conclusion of the project. The Binary Tree was the data structure that I used first to store all times that I max solar radiation had occurred, it was important to use there, because even if the file was not ordered by the correct times, because of the functions inorderTraversal() I was able to still display the times in order, as well as to keep unique registers. My second use of the binary tree, was to avoid double registers in my vector of metadata as mentioned before, and thirdly I use a binary tree of strings to display the menu in order based on the number of a function. I considered this last use trivial, but I think should have mentioned anyway.
On the other hand I have used the map only to allow the functionality of the last option of the menu, so what I did was while I was inserting the metadata object to a vector I also inserted in a map of metadata which I used to calculate the max solar radiations for a date.

Calculate max wind speed for year and month

To achieve this, I loop over all the elements of the vector and check the year of the register, then I have an Array of double that store the total wind speed, and the index of the array is the month of the object metadata.

- Calculate Average wind speed for year
- Don't add registers that have value zero to the calculation of average wind speed for a year
Again I loop over all elements of the array, checking for same year given by the user, then I check if the register is different of zero, otherwise, it will not count as zero is not relevant for the average. I created Array to store number of days and month, so this arrays of double will be used to make the calculations.

- Calculate Total solar radiation for year
- Don't add registers that have value lesser that 100 to the calculation of Solar Radiation for an year.
Similar to the previous one, in this case I loop all register, check for log greater than 100, use arrays of doubles to make calculations and display result for the user.

Save into a file the Average wind speed + solar radiation for year
No mystery here as both methods are explained above, the only difference is that instead of printing out in the console I would print in a file using ostream.

- Calculate Max solar radiation in a day
- Display all times that certain max solar radiation occurred in a given day.

In this option my approach was different of the other options, as here I loop over a map to find the highest solar radiation for the date. After I encounter the result, I would loop again over the map to find the time frame that the temperature occurred for then store it in a binary search tree.

Be able to exit program
Just use a break to go out of the loop and terminate the program.
Don't add registers that have value zero to the calculation of average wind speed for a year

Pros

- The pros of using vector and map as data structures is that it makes it easier to code, just looping all over the elements, in a force brute style.
- The binary tree was perfect to store the times and display them in order.

Cons

- My code could be faster if I had used BST as data structure for store the registers.
- If I was reading a really huge amount of data my code could become as slow due the amount of times that I loop over all the elements of the Vector.

Classes description

BinaryTree.h
Used as data structure has an optimized mechanism to search elements.

Date.h
Used to handle the dates given in the register.

Metadata.h
Works a object to hold the different logs of a register.

Time.h
Used to handle the time of a register.

Vector
Data structure that works like an array, however it scales its size according to its need. Largely used in my project.

Weather. H

This class was where I did all required calculations. It is here that I declare all my private data structures such as array, binary tree, map, vector and other elements.