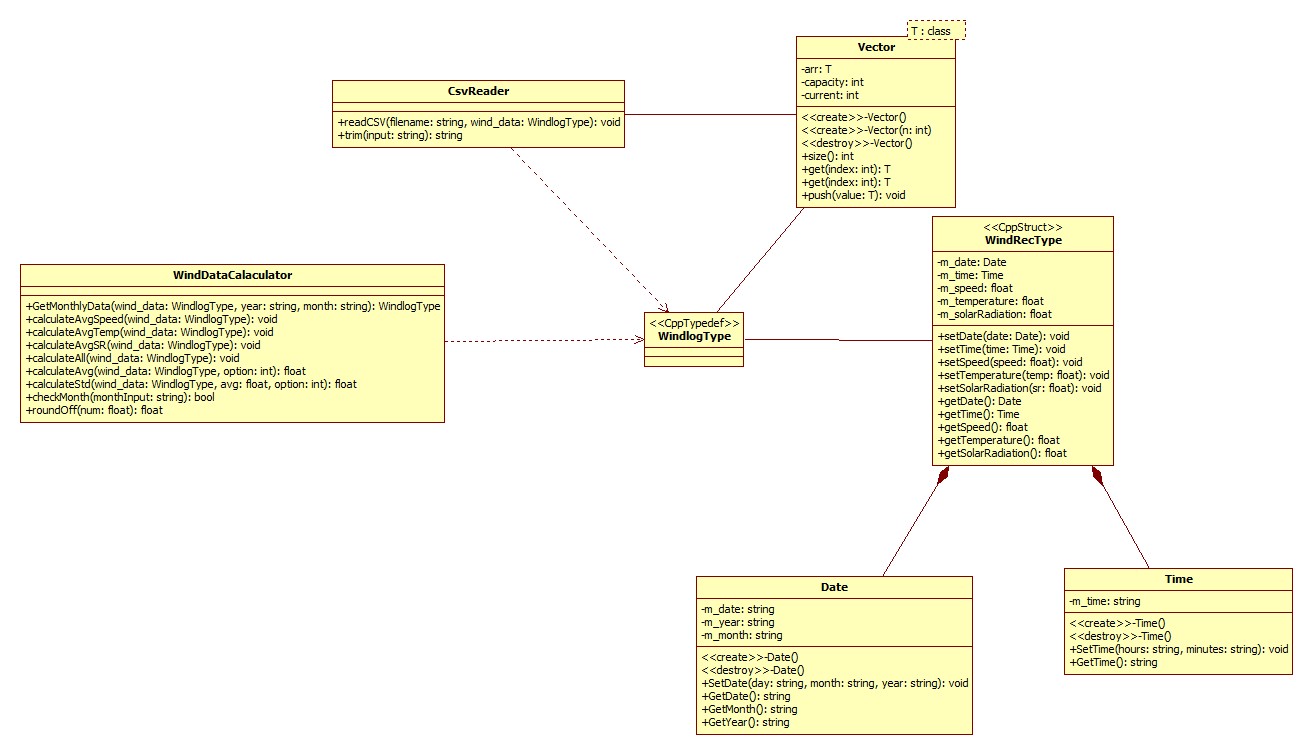
**UML Diagram**

Wind Recorder Caculator





Data Dictionary

|  |  |  |  |
| --- | --- | --- | --- |
| Function/  Member | Description | Type | Protection |
| WindRecType | Hold values such as Speed, Temperature, Solar Radiation, Date and time | Class |  |
| setDate(const Date& date) | Function to set the date and store it into m\_date | Void | + |
| setTime(const Time& time) | Function to set the time and store it into m\_time | void | + |
| |  | | --- | |  |   setSpeed(float speed) | Function to set the speed value and store it into m\_speed | void | + |
| setTemperature(float temp) | Function to set the Temperature value and store it into m\_temperature | void | + |
| setSolarRadiation(float sr) | Function to set the Solar Radiation value and store it into m\_solarRadiation | void | + |
| getDate() | Function to retrieve and access the date stored in m\_date | Date | + |
| getTime() | Function to retrieve and access the time stored in m\_time | Time | + |
| getSpeed() | Function to retrieve the speed value stored in m\_speed | float | + |
| getTemperature() | Function to retrieve the temperature value stored in m\_temp | float | + |
| getSolarRadiation() | Function to retrieve the solar radiation value stored in m\_solarRadiation | float | + |
| m\_date() | Variable to store the date of the wind record | Date | - |
| m\_time() | Variable to store the time of the wind record | Time | - |
| m\_speed() | Variable to store the speed of the wind record | float | - |
| m\_temperature() | Variable to store the temperature of the wind record | float | - |
| m\_solarRadiation() | Variable to store the Solar Radiation of the wind record | float | - |

|  |  |  |  |
| --- | --- | --- | --- |
| Function/  Member | Description | Type | Protection |
| WindlogType | Typedef for Vector<WindRecType>, representing a collection of wind records. |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Function/  Member | Description | Type | Protection |
| WindDataCalculator | Handles all the calculation process | Class |  |
| GetMonthlyData(const WindlogType& wind\_data, const string& year, const string& month) | This function creates a new instance of WindLogType and adds in data from the wind\_data taken from the function paremeter that matches the month and year also given in the month and year parameter. And returns WindlogType that only contains the records that matches the year and month parameters. | WindlogType | + |
| calculateAvgSpeed(const WindlogType& wind\_data) | This function calculates the average speed stored in the wind\_data record that matches the year and month entered by the user. And prints out the record for that specific month and year. | void | + |
| calculateAvgTemp(const WindlogType& wind\_data) | This function calculates the average temperature for all the months of a specific year stored in the wind\_data record. And prints out the average temperature for each of the specific year entered by the user. | void | + |
| const WindlogType& wind\_data | This function calculates the average solar radiation for all the months of a specific year stored in the wind\_data record. After calculating it converts the data from W/m**2** to kWh/m2 And prints out the average solar radiation in kWh/m2 for each months of the specific year entered by the user. | void | + |
| calculateAll(const WindlogType& wind\_data) | This function asks the user for a year. Then it calculates the wind speed temperature, solar radiation for each month of the year the user entered. It will then print it out into a csv file. | void | + |
| calculateAvg(const WindlogType& wind\_data, int option) | This function calculates the average of either wind speed, temperature, or solar radiation. The type of information to be calculated depends on the second parameter, int option. It then returns the result. | float | + |
| calculateStd(const WindlogType& wind\_data, int option) | his function calculates the standard deviation of either wind speed or temperature. The type of information to be calculated depends on the second parameter, int option. It then returns the result. | float | + |
| checkMonth(string& monthInput) | This function checks if the month that user enters is a correct month. It also converts the months with small case letters at the start of the string into the proper format that the program can understand | bool | + |
| roundOff(float num) | This function rounds off a number and returns the rounded off number | float | + |

|  |  |  |  |
| --- | --- | --- | --- |
| Function/  Member | Description | Type | Protection |
| CsvReader | Reads the csv file | Class | + |
| readCSV(const string& filename, WindlogType& wind\_data) | This function reads the csv file. It then determines which column contains the date, time, speed, temperature, solar radiation. It stores all the data into a vector and only retrieves the needed data into wind\_data. | void | + |
| trim(const string& input) | This function trims/ remove any whitespace, newline, tab in a string and returns the string. | string | + |

|  |  |  |  |
| --- | --- | --- | --- |
| Function/  Member | Description | Type | Protection |
| Vector | A resizable dynamic array | class | + |
| Vector() | Default Constructor |  | + |
| Vector(int n) | Default Constructor with initial capacity. |  | + |
| ~Vector() | Destructor |  | + |
| size() | This function Returns the number of elements in the vector | int | + |
| get(int index) | This function Retrieves the current element specified by the parameter index | T& | + |
| push(const T& value) | This function Add another element into the end of the vector | void | + |

Date

|  |  |  |  |
| --- | --- | --- | --- |
| Function/  Member | Description | Type | Protection |
| SetDate(string day, string month, string year) | This function Sets the date into a full string. It also sets the m\_date, m\_year, m\_month. | void | + |
| GetDate() | This function returns the m\_date which contains the full date string | string | + |
| GetMonth() | This function returns the m\_month which contains the month. | string | + |
| GetYear() | This function returns the m\_year which contains the year. | string | + |
| m\_date | A Variable to store the full date string | string | - |
| m\_month | A Variable to store the month | string | - |
| m\_year | Variable to store the year | string | - |

Time

|  |  |  |  |
| --- | --- | --- | --- |
| Function/  Member | Description | Type | Protection |
| SetTime(string hours, string minutes) | This function sets the m\_time | Void | + |
| GetTime | This function returns the m\_year which contains the time. | String | + |
| m\_time | A Variable to store the time |  | - |

**Rationale**

The reason why I designed my program this way is to make my code modular and more maintainable. I made a header file for WindRecType struct so that the other classes I made can gain access to the setters and getters of my struct. The reason behind why I made the CsvReader header and source file is so that, if I were to expand my code and made other classes that will need access to reading a csv file into a struct, I can just reuse the code. The reason why I made WindDataCalculator into its own class is to make my code more cleaner, and when there is a problem with the calculations, I know exactly which part of my code needs to be looked at. I implemented the previous lesson’s Date and Time classes, to store and segregate Date and time values, such as what I did with the Date class. I implemented the previous lab exercise’s Vector class that I made myself, because I have more control on the data, and I can customize it to optimize for specific usage

**Algorithm**The following are algorithms which are essential to this program. The variables are different from the actual program.

**getmonthlydata**

//allData will be a parameter value

//monthInput and yearInput will be a parameter value

WindData monthData //create a new record

For (i=0; i < allData.size; i++)

if (allData(i).year == yearInput and allData(i).Month = monthInput)

monthData(i) = allData(i)

return monthData

**calculateavg**

sum = 0

for (i = 0; i < wind\_data.size; i++ )

sum = sum + wind\_data[i].speed/temperature/solar radiation

average = sum / wind\_data.size

return average

**calculatestd**

avg; // this is from a parameter with value

float standardDeviation = 0.0f ;

for (i = 0; i < wind\_data.size ; i++)

standardDeviation += (wind\_data[i].speed – avg)^2

ENDfor

standardDeviation = sqrt(standardDeviation / wind\_data.size)

return standardDeviation

**calculatespeed/temp/sr**

monthArray[] // assume all the months of the year are stored here

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

//Create new instance of WindlogType and retrieve the values of the month and year the user entered

WindlogType monthlyData = GetMonthlyData(currentWindData, month[i], year)

avg = calculateavg(monthlyData, 1) //the second parameter controls //which wind record data to calculate the average. For this example I want to //get the speed so I will it to 1

std = calculatestd(monthlyData, 1)// same as the function above

if (avg != 0)

Display the month and month’s average and/or standard deviation

**readcsv**

//Declare and initialize a vector string called header

//Declare and initialize a vector float called sensor

string temp = “ ”;

OpenFile (filename)

Read(filename)  
while (next string is not end of line)

temp = insert (a string seperated by “,”)

push.(temp)

temp = “ ”;

ENDwhile

int speed\_index = 0;

int temperature\_index = 0;

int solarRadiation\_index = 0;

for (i=0; i < header.size: i++)

if header.get(i) == S

speed index = i

if header.get(i) == T

temperature index = i

if header.get(i) == SR

solarRadiation index = i

ENDfor

while(next line is not end of file)

float sensorTemp

while (next string is not end of line)

temp = insert (a string seperated by “,”)

sensorTemp = push.changeToFloat

sensor.push(sensorTemp)

sensorTemp = 0.0f

temp = “ ”

ENDwhile

TempWindData //create new instance of wind data record

// date is always at the start of the column

TempWindData date = sensor.get(0)

//same reason as date

TempWindData time = sensor.get(1)

//Get the position of the sensor by using the index we created earlier

TempWindData speed = sensor.get(speed index)

TempWindData temperature = sensor.get(temperature index)

TempWindData solar radiation = sensor.get(solar radiation index)

WindData = TempWindData

Close file

**vector push**

//ArrayCurrent is a member variable and contains the current array

//Value is a parameter that will be pushed into the back of the array

if(currentsize == maxsize)

new ArrayTemp

ArrayTemp size = 2 \* maxsize

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

ArrayTemp[i] = ArrayCurrent [i]

Delete ArrayCurrent

maxsize += 2

ArrayCurrent = ArrayTemp

ENDfor

ENDif

ArrayCurrent[value]

currentsize++

**TEST PLAN**

To test if our program is working we will use the default csv filepath located in data\_source.txt which is the MetData\_Mar01-2015-Mar01-2016-ALL.csv. To ensure results are accurate and consistent I will be only using data from the year 2015. I will be testing each function of the program.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test id** | **Test description/justification** | **Actual data for this test** | **Expected output** | **Actual program output when test is carried out** | **Test run outcome – Pass/Fail** |
| 1 | Check if option 1 Calculate Average wind speed is working | User Enters 1 to access first option.  User Answers  “March” for the Month Prompt  User Answers  “2015” for the Year Prompt | March 2015 Average: xxxxk/m2  Ssample stdev: xxxx | Refer to figure 1 below | Pass |
| 2 | Check if option 2 Calculate Average wind temperature is working | User Enters 2 to access second option.  User Answers  “2015” for the  year prompt | Print out the year.  Below the year it prints out the average temperature and stdev for each month | Refer to figure 2 below | Pass |
| 3 | Check if option 3 Calculate Average wind solar radiation is working | User Enters 3 to access third option.  User Answers 2015 for the year prompt | Print out the year  Below the year it prints out the average solar radiation for each month. | Refer to figure 3 below | Pass |
| 4 | Check if option 4  Calculate Average wind speed, temperature, radiation is working | User Enters 4 to access fourth option.  User Answers 2015 for the year prompt | It should go back to the option menu and a WindTempSolar.csv file should be made and filled up with data | Refer to figure 4 below | Pass |
| 5 | Check if option 5  Exits properly | User Enters 5 | It should return 0 | Refer to figure 5 below | Pass |

Output of Test runs

Figure1. TestID 1

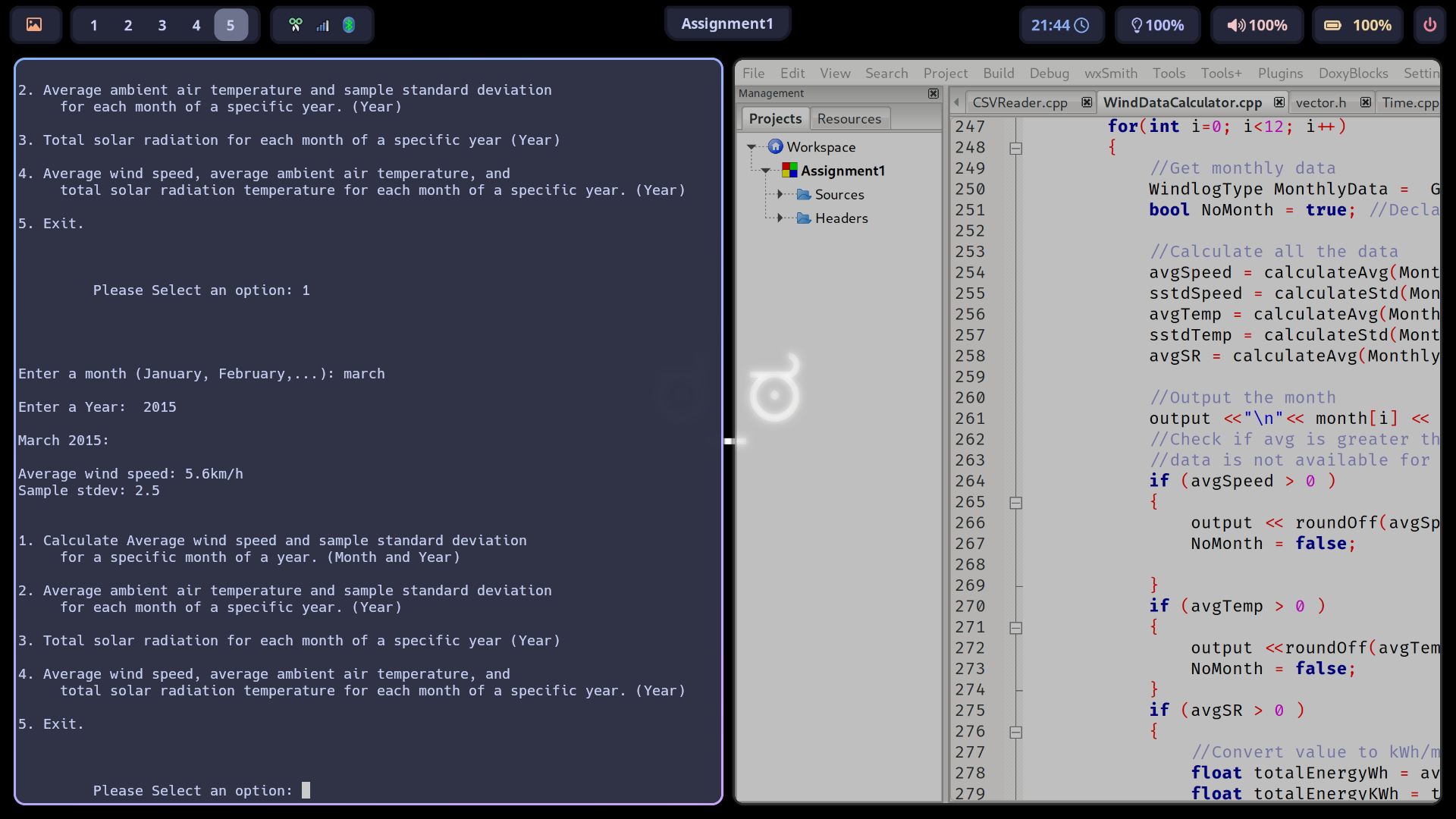


Figure2 TestID 2

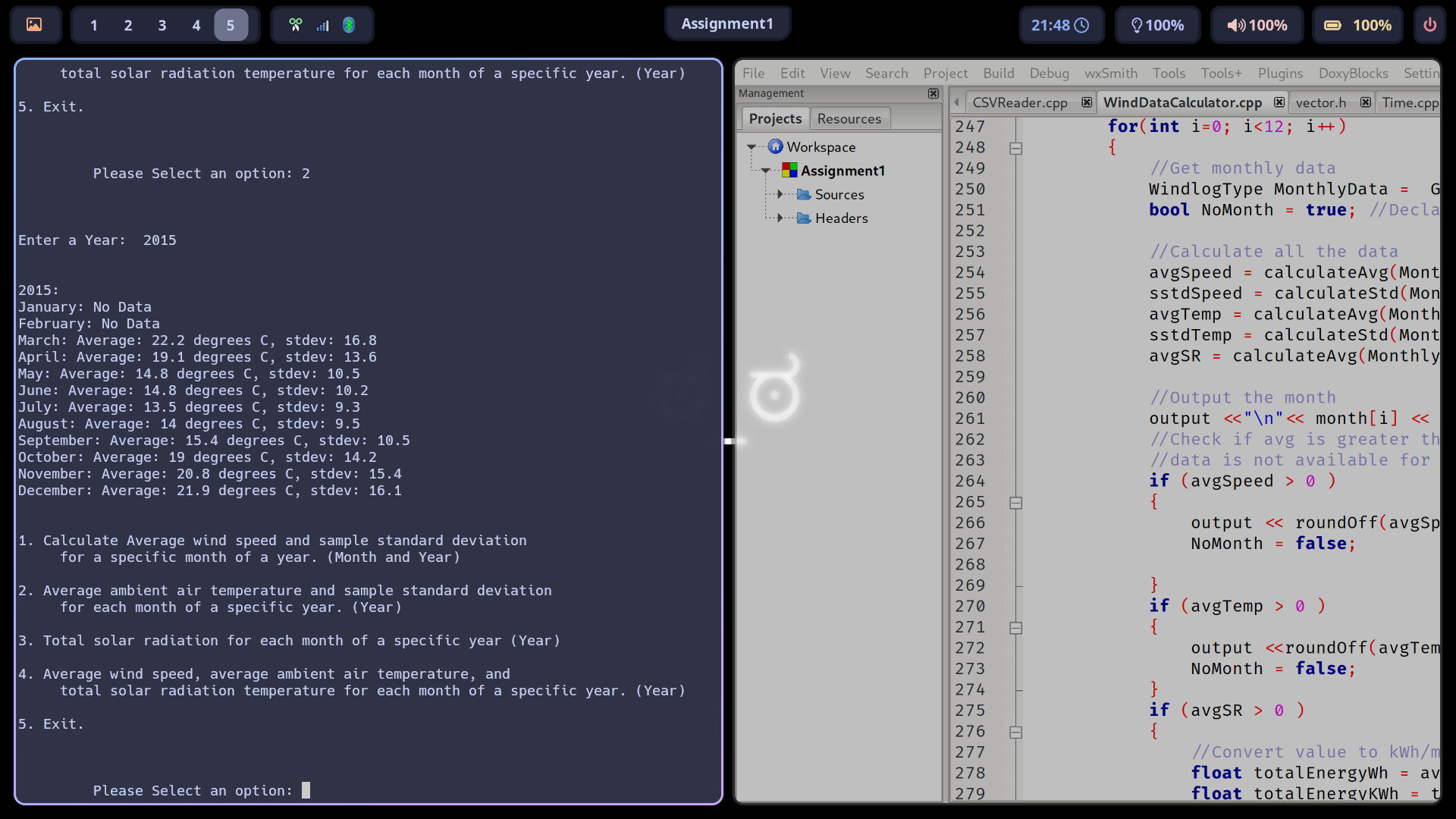


Figure3. TestID 3

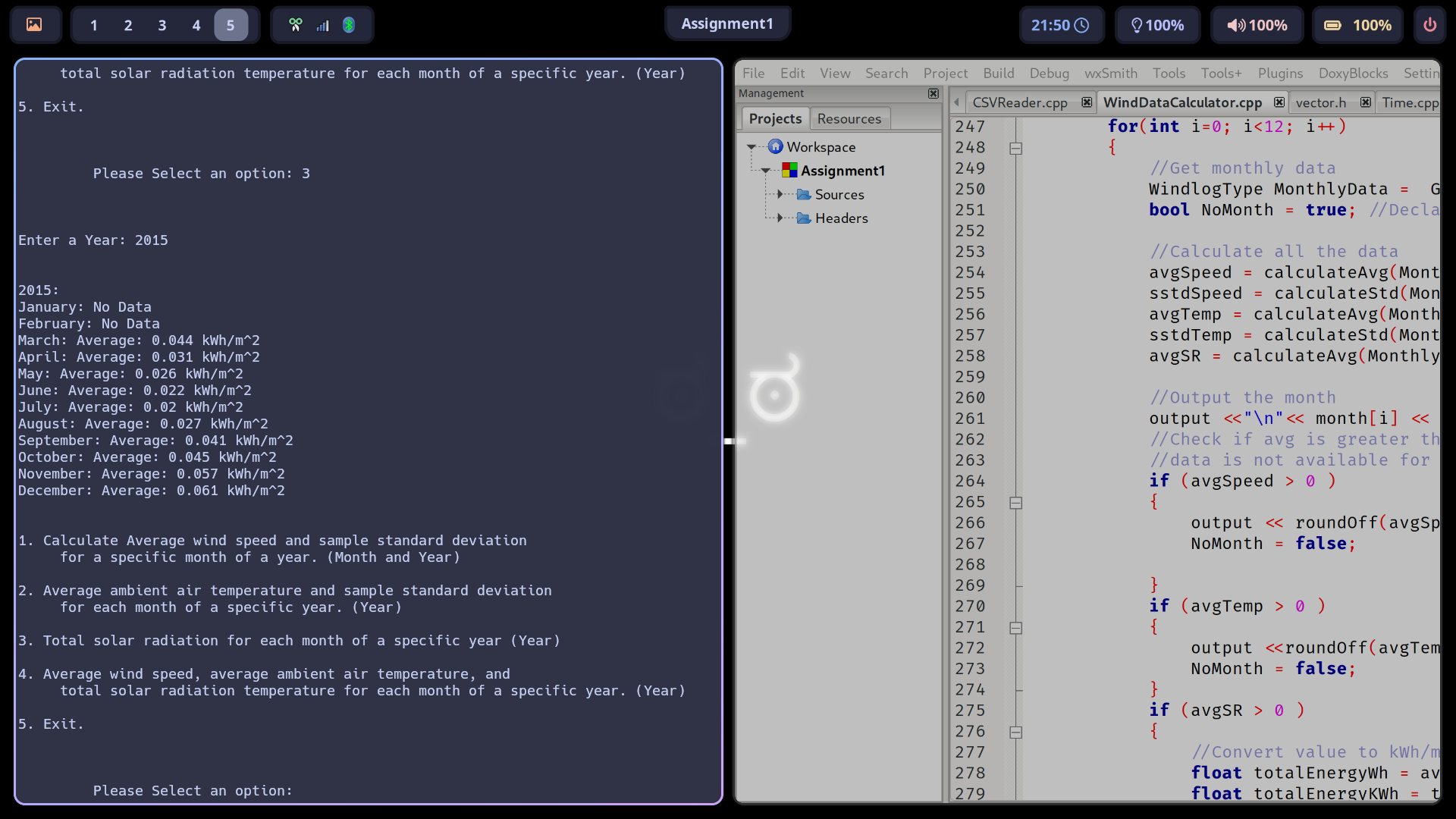


Figure 4. TestID 4



Figure 5. TestID 5

