Contents

[Data Dictionary 3](#_Toc146582527)

[Date Class 3](#_Toc146582528)

[Time Class 4](#_Toc146582529)

[WindLogType Class 5](#_Toc146582530)

[Vector Class 6](#_Toc146582531)

[Utils Class 7](#_Toc146582532)

[DataReader Class 8](#_Toc146582533)

[Converter Class 8](#_Toc146582534)

[Menu Class 9](#_Toc146582535)

[Test Plan & Unit Test 10](#_Toc146582536)

[Date Class Test 10](#_Toc146582537)

[Time Class Test 10](#_Toc146582538)

[WindLogType Class Test 10](#_Toc146582539)

[Vector Class Test 11](#_Toc146582540)

[Converter Class Test 12](#_Toc146582541)

[DataReader Class Test 13](#_Toc146582542)

[Algorithm 13](#_Toc146582543)

[DataReader 13](#_Toc146582544)

[Converter 14](#_Toc146582545)

[Vector 15](#_Toc146582546)

[Date 18](#_Toc146582547)

[Time 20](#_Toc146582548)

[WindLogType 22](#_Toc146582549)

[Menu 25](#_Toc146582550)

[Utils 29](#_Toc146582551)

# Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date Class | | | | |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| m\_day | Member Variable | Private (-) | Storing the unsigned data type day of a date | Used to keep track of the Date of a day |
| m\_month | Member Variable | Private (-) | Storing the unsigned data type month of a date | Used to keep track of the Date of a month |
| m\_year | Member Variable | Private (-) | Storing the unsigned data type year of a date | Used to keep track of the Date of a year |
| Date() | Default Constructor | Public (+) | Holding the day, month, year as to be set as 0 | Used to construct a default Date object once initialized |
| Date(const unsigned day, const unsigned month, const unsigned year) | User defined Parameterized Constructor | Public (+) | Holding the day, month, and year of user defined value by passing in as parameter. | Used to construct a parameterized Date object with user defined values once initialized. |
| unsigned GetDay() const | Member function | Public (+) | Returns an unsigned day of a date. | Used to retrieve the day from the date class object. |
| unsigned GetMonth() const | Member function | Public (+) | Returns an unsigned month of a date. | Used to retrieve the month from the date class object. |
| unsigned GetYear() const | Member function | Public (+) | Returns an unsigned year of a date. | Used to retrieve the year from the date class object. |
| void SetDay(const unsigned day) | Procedure | Public (+) | Setting user defined day by passing it as a parameter | Used to set the day of the Date class object |
| void SetMonth(const unsigned month) | Procedure | Public (+) | Setting user defined month by passing it as a parameter | Used to set the month of the Date class object |
| void SetYear(const unsigned year) | Procedure | Public (+) | Setting user defined year by passing it as a parameter | Used to set the year of the Date class object |
| ostream &operator << (ostream &outputStream, const Date &D) | Overloaded output stream | Public (+) | Overloads the Output operator to the format as user defined | Used to output user defined output format |
| istream &operator >> (istream &inputStream, Date &D) | Overloaded input stream | Public (+) | Overloads the input operator in reading user defined format | Uset to input / read user defined input format |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Time Class | | | | |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| m\_hour | Member Variable | Private (-) | Storing the string data type hour of a time | Used to keep track of the hour of a time |
| m\_minute | Member Variable | Private (-) | Storing the string data type minute of a time | Used to keep track of the minute of a time |
| Time() | Default Constructor | Public (+) | Holding the day, month, year as to be set as 0 | Used to construct a default Date object once initialized |
| Time(const string &hour, const string &minute) | User defined Parameterized Constructor | Public (+) | Holding the hour, and minute of user defined time by passing in as parameter. | Used to construct a parameterized Time object with user defined time once initialized. |
| unsigned GetHour() const | Member function | Public (+) | Returns a string hour of a date. | Used to retrieve the hour from the time class object. |
| unsigned GetMinute() const | Member function | Public (+) | Returns a string minute of a date. | Used to retrieve the minute from the time class object. |
| void SetHour(const string &hour) | Procedure | Public (+) | Setting user defined hour by passing it as a parameter | Used to set the hour of the Time class object |
| void SetMinute(const string &minute) | Procedure | Public (+) | Setting user defined minute by passing it as a parameter | Used to set the minute of the Time class object |
| ostream &operator << (ostream &outputStream, const Time &T) | Overloaded output stream | Public (+) | Overloads the Output operator to the format as user defined | Used to output user defined output format |
| istream &operator >> (istream &inputStream, Time &T) | Overloaded input stream | Public (+) | Overloads the input operator in reading user defined format | Uset to input / read user defined input format |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| WindLogType Class | | | | |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| Date m\_date | Member Variable | Private (-) | Storing the Date class data type of a date object | Used to keep track of the date object |
| Time m\_time | Member Variable | Private (-) | Storing the Time class data type of a time object | Used to keep track of the time object |
| m\_windspeed | Member Variable | Private (-) | Storing the float data type of the windspeed | Used to keep track of the windspeed |
| m\_airTemperature | Member Variable | Private (-) | Storing the float data type of the air temperature | Used to keep track of the air temperature |
| m\_solarRadiation | Member Variable | Private (-) | Storing the float data type of the solar radiation | Used to keep track of the solar radiation |
| WindLogType() | Default Constructor | Public (+) | Holding the date object, time object, windspeed, air temperature, and solar radiation as to be set as 0 | Used to construct a default WindLogType object once initialized |
| WindLogType(const Date &date, const Time &time, const float windspeed, const float airTemperature, const float solarRadiation) | User defined Parameterized Constructor | Public (+) | Holding the user define parameterized date object, time object, windspeed, air temperature, and solar radiation | Used to construct a user defined values as parameters to the WindLogType object once initialized |
| const Date GetDate() const | Member function | Public (+) | Returns a date object of a date. | Used to retrieve the date object |
| const Date GetTime() const | Member function | Public (+) | Returns a time object of a date. | Used to retrieve the time object |
| const float GetWindSpeed() const | Member function | Public (+) | Returns a float windspeed value | Used to retrieve the windspeed from the WindLogType class object. |
| const float GetAirTemperature() const | Member function | Public (+) | Returns a float air temperature value | Used to retrieve the air temperature from the WindLogType class object. |
| const float GetSolarRadiation() const | Member function | Public (+) | Returns a float solar radiation value | Used to retrieve the solar radiation from the WindLogType class object. |
| void SetDate(const Date &date) | Procedure | Public (+) | Setting user defined date by passing it as a parameter | Used to set the date of the Date class object |
| void SetTime(const Time &time) | Procedure | Public (+) | Setting user defined time by passing it as a parameter | Used to set the time of the Time class object |
| void SetWindSpeed(const float windspeed) | Procedure | Public (+) | Setting user defined windspeed by passing it as a parameter | Used to set the windspeed of the WindLogType class object |
| void SetAirTemperature(const float airTemperature) | Procedure | Public (+) | Setting user defined air temperature by passing it as a parameter | Used to set the air temperature of the WindLogType class object |
| void SetSolarRadiation(const float windspeed) | Procedure | Public (+) | Setting user defined solar radiation by passing it as a parameter | Used to set the solar radiation of the WindLogType class object |
| ostream &operator << (ostream &outputStream, const WindLogType &W) | Overloaded output stream | Public (+) | Overloads the Output operator to the format as user defined | Used to output user defined output format |
| istream &operator >> (istream &inputStream, WindLogType &W) | Overloaded input stream | Public (+) | Overloads the input operator in reading user defined format | Uset to input / read user defined input format |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vector Class | | | | |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| m\_capacity | Member Variable | Private (-) | Storing the unsigned data type capacity of a dynamic array in the heap | Used to keep track of the capacity of the dynamic array |
| m\_used | Member Variable | Private (-) | Storing the unsigned data type of a dynamic array used space in the heap | Used to keep track of the used space of the dynamic array |
| m\_array | Member Variable | Private (-) | Storing the T pointer data type of the address of a dynamic array in the heap | Used to keep track of the pointer address pointing at the dynamic array |
| unsigned GetCapacity() const | Member function | Public (+) | Returns the capacity of a dynamic array. | Used to retrieve the capacity of a dynamic array. |
| unsigned GetUsed() const | Member function | Public (+) | Returns the amount of been used space of a dynamic array. | Used to retrieve the number of used spaces in a dynamic array. |
| T \*GetArray() const | Member function | Public (+) | Returns the address of a dynamic array that the pointer is pointing at | Used to retrieve the address of the pointer pointing in a dynamic array. |
| Vector(const Vector &initializer) | Procedure | Public (+) | Deep copying an existing vector object by passing in as a parameter to this new vector creation | Used to deep copy a newly created vector from an existing vector |
| Vector &operator = (const Vector &rhs) | Overload equals operator | Public (+) | Returns \*this values from the deep copied rhs existing vector object | Used to retrieve the values copied from the rhs existing vector object |
| bool Append(const T &value) | Member function | Public (+) | Returns true when T data type reference value has been added into the dynamic array | Used to append the T data type reference value into the dynamic array |
| T &GetValueByIndex(unsigned index) const | Member function | Public (+) | Returns the reference value of the value from the index of the dynamic array | Used to return the value that is contained in the dynamic index |
| Void Resize() | Member function | Private (-) | This doubles the dynamic array capacity when used slot is half of the size of the capacity | Used for resizing the allocation of the dynamic array capacity |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Utils Class | | | | |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| string GetMonthInStr(const unsigned &month) | Member Variable | Public (+) | Return the string of the month by passing in the numeric number of the month | Used to convert the month in string |
| void assert(bool condition, string passMsg, string failMsg, ostream& output, unsigned &total Test, unsigned &testPassed, unsigned &testFailed); | Member Variable | Public (+) | Used for testing the condition on whether the test pass or fail, and outputs the result to a txt file | Used for testing purposes only |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DataReader Class | | | | |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| m\_filename | Member Variable | Private (-) | Storing the filename of a file | Used to keep track of the filename |
| DataReader(const string &filename) | User defined parameterized constructor | Public (+) | Holding the filename used in the application | Used to read the pass in filename from the path |
| bool ReadAppendData(Vector<WindLogType> &windlogVector) | Member function | Public (+) | Holding the vector data type for appending and reading the data from the filename | Used to read the data and append the data from to the vector which has been passed in. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Converter Class | | | | |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| const float GetAverageWindSpeed(float sum, unsigned count) | Member Function | Public (+) | Returns the float data type of getting the average windspeed by passing in the total float data type sum and unsigned data type count | Used to convert and calculate the average windspeed |
| const float GetAverageAirTemperature(float sum, unsigned count) | Member Function | Public (+) | Returns the float data type of getting the average air temperature by passing in the total float data type sum and unsigned data type count | Used to convert and calculate the average air temperature |
| const float GetSolarRadiationTotal(float sum) | Member Function | Public (+) | Returns the float data type of getting the average windspeed by passing in the total float data type sum and unsigned data type count | Used to convert the total sum of solar radiation |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Menu Class | | | | |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| static unsigned RunMenu(Vector<WindLogType>& windLogVector) | Member Function | Public (+) | Returns the unsigned data type choice after user selection, displaying different option of what the user selected | Used to return the numeric unsigned int of user selection to the switch case of different display |
| static void OptionOne(Vector<WindLogType>& windLogVector); | Member Function | Public (+) | Takes in user’s month and year in looking through the pass in vector parameter. And prints out the average windspeed and average air temperature which is equals to user’s selection of month and year | Used to find the average windspeed and air temperature of a specific month and year from user input |
| static void OptionTwo(Vector<WindLogType>& windLogVector); | Member Function | Public (+) | Takes in user’s year in looking through the pass in vector parameter. And prints a list from Jan’s to Dec’s average windspeed and air temperature | Used to display a list of average windspeed and air temperature of a specific year from user input |
| static void OptionThree(Vector<WindLogType>& windLogVector); | Member Function | Public (+) | Takes in user’s year in looking through the pass in vector parameter. And prints a list from Jan’s to Dec’s total sum of solar radiation | Used to display a list of solar radiation of a specific year from user input |
| static void OptionFour(Vector<WindLogType>& windLogVector); | Member Function | Public (+) | Takes in user’s year in looking through the pass in vector parameter. And outputs to a csv file a list of average windspeed, average air temperature, and the total sum of solar radiation for each month | Used to output a csv file of a list for windspeed, air temperature, and solar radiation of a specific year from user input |

# Test Plan & Unit Test

Unit test report in txt, available in the folder report folder.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Date Class Test | | | | |
| Description | Change Code | Actual Output | Expected Output | Status |
| Test on default constructor and overloaded output stream | Date date;  cout << date; | 0/0/0 | 0/0/0 | Pass |
| Test on setting day of a date | date.SetDay(21) | 21 | 21 | Pass |
| Test on setting month of a date | date.SetMonth(7) | 7 | 7 | Pass |
| Test on setting year of a date | date.SetYear(2020) | 2020 | 2020 | Pass |
| Test on converting month in string from a numeric number of a month | date.SetMinthInStr(date.GetMonth()) | July | July | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Time Class Test | | | | |
| Description | Change Code | Actual Output | Expected Output | Status |
| Test on default constructor and overloaded output stream | Time time;  cout << time; | 0:00 | 0:00 | Pass |
| Test on setting hour of a time | time.SetHour(12) | 12 | 12 | Pass |
| Test on setting minute of a time | time.SetMinute(22) | 22 | 22 | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| WindLogType Class Test | | | | |
| Description | Change Code | Actual Output | Expected Output | Status |
| Test on default constructor and overloaded output stream | WindLogType defaultWindLogType;  cout << defaultWindLogType; | 0/0/0 0:00 0 0 0 | 0:00 | Pass |
| Test on setting a date with parameterized constructor | Date setDate(day, month, year)  defaultWindLogType.SetDate(setDate) | 11/12/2020 | 11/12/2020 | Pass |
| Test on setting a time with parameterized constructor | Time setTime(hour, minute)  defaultWindLogType.SetTime(setTime) | 11:12 | 11:12 | Pass |
| Test on setting a windspeed | defaultWindLogType.SetWindSpeed(29.1) | 29.1 | 29.1 | Pass |
| Test on setting an air temperature | defaultWindLogType.SetAirTemperature(29.1) | 32.3 | 32.3 | Pass |
| Test on setting an solar radiation | defaultWindLogType.SetSolarRadiation(43.1) | 43.1 | 43.1 | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Vector Class Test | | | | |
| Description | Change Code | Actual | Expected | Status |
| Test on default constructor  (unsigned data type) | Vector<unsigned>unsignedVec;  typeid(unsignedVec).name() == typeid(Vector<unsigned>).name() | true | true | Pass |
| Test on default constructor  (string data type) | Vector<string>stringVec;  typeid(stringVec).name() == typeid(Vector<string>).name() | true | true | Pass |
| Test on default constructor  (Date object data type) | Vector<Date>dateVec;  typeid(dateVec).name() == typeid(Vector<Date>).name() | true | true | Pass |
| Test on default constructor  (Time object data type) | Vector<Time>timeVec;  typeid(timeVec).name() == typeid(Vector<Time>).name() | true | true | Pass |
| Test on default constructor  (WindLogType object data type) | Vector<Time>windLogVec;  typeid(windLogVec).name() == typeid(Vector<WindLogType>).name() | true | true | Pass |
| Test on default constructed object initial capacity | unsignedVec.GetCapacity() | 2 | 2 | Pass |
| Test on default constructed object initial used allocated space in the dynamic array | unsignedVec.GetUsed() | 0 | 0 | Pass |
| Test on default constructed object pointing to the 1st index of the allocated dynamic array which is not a nullptr | unsignedVec.GetArray() != nullptr | true | true | Pass |
| Test on appending a value to the vector | unsignedVec.Append(12) == true | true | true | Pass |
| Test on getting the value from the 1st index from the dynamic array | unsignedVec.GetValueByIndex(0) | 12 | 12 | Pass |
| Test on Copy Constructor  (To check on the capacity, used slot, and the value in the index are the same. Checking that the array pointer is not pointing at the same dynamic array.) | Vector<unsigned>anotherUnsignedVector(unsignedVec);   1. anotherUnsignedVec.GetCapacity() == unsignedVec.GetCapacity(); 2. anotherUnsignedVec.GetUsed() == unsignedVec.GetUsed(); 3. anotherUnsignedVec.GetValueByIndex(0) == unsignedVec.GetValueByIndex(0); 4. anotherUnsignedVec.GetArray() != unsignedVec.GetArray(); 5. &anotherUnsignedVec.GetValueByIndex(0) != unsignedVec.GetValueByIndex(0); | 1. true 2. true 3. true 4. true 5. true | 1. true 2. true 3. true 4. true 5. true | Pass |
| Test on Overload equals operator  (To check on the capacity, used slot, and the value in the index are the same. Checking that the array pointer is not pointing at the same dynamic array.) | Vector<unsigned>thirdUnsignedVec;  thirdUnsignedVec = unsignedVec;   1. thirdUnsignedVec.GetCapacity() == unsignedVec.GetCapacity(); 2. thirdUnsignedVec.GetUsed() == unsignedVec.GetUsed(); 3. thirdUnsignedVec.GetValueByIndex(0) == unsignedVec.GetValueByIndex(0); 4. thirdUnsignedVec.GetArray() != unsignedVec.GetArray(); 5. & thirdUnsignedVec.GetValueByIndex(0) != unsignedVec.GetValueByIndex(0); | 1. true 2. true 3. true 4. true 5. true | 1. true 2. true 3. true 4. true 5. true | Pass |
| Scenario Test:   1. To append 2 elements into a vector. 2. Add and update allocated capacity. 3. Update the used spaces in the dynamic array. | Vector<unsigned>scenarioVec;  scenarioVec.Append(1);  scenarioVec.Append(2);   1. scenarioVec.GetCapacity() == 4; 2. scenarioVec.GetUsed() == 2; | 1. true 2. true | 1. true 2. true | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Converter Class Test | | | | |
| Description | Change Code | Actual | Expected | Status |
| Test on converting average wind speed | Converter converter;  converter.GetAverageWindSpeed(20.5, 2); | 36.9 | 36.9 | Pass |
| Test on converting average air temperature | Converter converter;  converter.GetAverageAirTemperature(20.5, 2); | 10.25 | 10.25 | Pass |
| Test on converting total solar radiation | Converter converter;  converter.GetSolarRadiation (12000); | 2 | 2 | Pass |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| DataReader Class Test | | | | |
| Description | Change Code | Actual | Expected | Status |
| Test on failure loading and reading CSV with the wrong filename | Vector<WindLogType> windLogVec;  DataReader wrongDataReader(windLogVec);  wrongDataReader.ReadAppendData(windLogVec); | false | false | Pass |
| Test on failure loading and reading CSV with the wrong filename | Vector<WindLogType> windLogVec;  DataReader rightDataReader(windLogVec);  rightDataReader.ReadAppendData(windLogVec); | true | true | Pass |

# Algorithm

|  |
| --- |
| DataReader |
| * ***bool ReadAppendData(Vector<WindLogType> &windlogType);***   Returns Boolean on Reading and storing the data from the CSV file from path / filename. Takes in a vector of WindLogType class data type.   * Initializing a windLogType constructor that would be used for the in-file stream. * Passing in the filename to ifstream. * Checks with condition that filename / path does not exists.   + Console out the error of file unable to open.   + Return false. * \* Skips the header row before istream the data. * While looping and checks the condition by peeking the lines do not equal to end of line.   + Istream in the windLogType constructed constructor.   + Append the istream(ed) data after reading to the vector. * Return true. |

|  |
| --- |
| Converter |
| * ***const float GetAverageWindSpeed(float sum, unsigned count);***   Returns float datatype on converting the average windspeed. Takes in the sum of the windspeed from CSV file and the total count of every added element.   * Converting average wind speed as 🡪 (sum / count) \* 3.6. * Return the calculated average wind speed. |
| * ***const float GetAverageAirTemperature(float sum, unsigned count);***   Returns float datatype on converting the average air temperature. Takes in the sum of the air temperature from CSV file and the total count of every added element.   * Calculate the average air temperature as 🡪 sum / count. * Return the calculated average air temperature. |
| * ***const float GetSolarRadiationTotal(float sum);***   Returns float datatype on calculating and converting the total sum of solar radiation . Takes in the sum of the solar radiation added from CSV file.   * Calculate the total sum of solar radiation as 🡪 sum / 6000. * Return the calculated converted sum of solar radiation. |

|  |
| --- |
| Vector |
| * ***Vector();***   Used for constructing a vector which sets all member variables to initial variable as 0.   * Sets m\_capacity to initial value of 2. * Sets m\_used to initial value of 0. * Creates a new array in the heap with initial value of 2 and sets to m\_array. |
| * ***~Vector();***   A destructor that deletes the dynamic array in the heap and pointing the array address to nullptr after deleting the dynamic array.   * Checks if the member variable m\_array is pointing to nullptr. * If NOT:   + Delete the dynamic array in the heap.   + Set member variable m\_array to nullptr. |
| * ***Vector(const Vector<T> &initializer);***   A copy constructor which takes in another vector as parameter. Copies the member variables to this initialized vector by using the ‘Copy’ method.   * + Passing in the ‘Copy’ method that takes in the initializer as parameter. |
| * ***Vector &operator=(const Vector<T> &rhs);***   An overload equals operator that takes in another vector alias as rhs and copies the current variables to another vector alias rhs.   * Checks if this constructing address of the vector is the same as the address of the alias rhs address. * If NOT:   + Delete the dynamic array in the heap.   + Use the copy method and pass in the alias vector rhs. * Return the de-reference value of this vector. |
| * ***void GetCapacity() const.***   Used for getting the capacity of the dynamic array from the member variable m\_capacity.   * + Returns the member variable m\_capacity. |
| * ***void GetUsed() const.***   Used for getting the used slot of the dynamic array from the member variable m\_used.   * + Returns the member variable m\_used. |
| * ***T \*GetArray() const.***   Used for getting the address of the dynamic array from the member variable m\_array.   * + Returns the member variable address m\_array. |
| * ***bool Append(const T &value).***   Used for appending element into a vector.   * + Checks if used slot is equals half of the dynamic array capacity.   + If YES:     - Use the Resize method to expand the array.   + Sets the value to member variable[member of used slot].   + Increment used slot by 1.   + Return true. |
| * ***T &GetValueByIndex(unsigned index) const***   Used to return element from a vector of a given index.   * Return member variable dynamic array of member variable index. |
| * ***void Resize()***   Used to resize the capacity of the dynamic array when used slot is half of the capacity.   * + Doubles the capacity of the array with a value.   + Creates a new array in the dynamic array.   + For loop:     - Sets the current array element to the newly created dynamic array with the doubled capacity.   + Deletes the current array after setting the element to the newly created array.   + Creates a new array for pointer member variable m\_array.   + For loop:     - Sets the new array elements to the member variable m\_array dynamic array.   + Deletes the new array after setting it back to the member variable created dynamic array. |

|  |
| --- |
| Date |
| * ***Date();***   Used for constructing a Date object which sets all member variables to initial variable as 0.   * Sets m\_day to initial value of 0. * Sets m\_month to initial value of 0. * Sets m\_year to initial value of 0. |
| * ***Date(const unsigned day, const unsigned month, const unsigned year);***   Used for constructing a Date object that uses parameterized value to be set into member variable day, month, year.   * Sets m\_day to parameterized value of the day. * Sets m\_month to parameterized value of the month. * Sets m\_year to parameterized value of the year. |
| * ***unsigned GetDay() const;***   Used to get a day from the Date object.   * Returns the member variable unsigned day. |
| * ***unsigned GetMonth() const;***   Used to get a month from the Date object.   * Returns the member variable unsigned month. |
| * ***unsigned GetYear() const;***   Used to get a year from the Date object.   * Returns the member variable unsigned year. |
| * ***void SetDay(const unsigned day);***   Used to set a day from the Date object.   * Sets the parameterized unsigned day to the member variable m\_day. |
| * ***void SetMonth(const unsigned month);***   Used to set a month from the Date object.   * Sets the parameterized unsigned month to the member variable m\_month. |
| * ***void SetYear(const unsigned year);***   Used to set a year from the Date object.   * Sets the parameterized unsigned year to the member variable m\_year. |
| * ***ostream &operator << (ostream &outputStream, const Date &D);***   returns the output stream user defined Date object.   * Gets the date, month, and year from and return the output stream. |
| * ***istream &operator << (istream &inputStream, const Date &D);***   returns the input stream user defined Date object.   * Sets and converts the string date, month and year to unsigned int from and return the input stream. |

|  |
| --- |
| Time |
| * ***Time();***   Used for constructing a Time object which sets all member variables to initial variable as 0.   * Sets m\_hour to initial value string of 0. * Sets m\_minute to initial value string of 00. |
| * ***Time(const string &hour, const string &minute);***   Used for constructing a Time object that uses parameterized reference value to be set into member variable hour, minute.   * Sets the reference hour to member variable value string m\_hour. * Sets the reference minute to member variable value string m\_minute. |
| * ***unsigned GetHour() const;***   Used to get an hour from the Time object.   * Returns the member variable string hour. |
| * ***unsigned GetMinute() const;***   Used to get a minute from the Time object.   * Returns the member variable string month. |
| * ***void SetHour(const string &hour);***   Used to set an hour reference from the Time object.   * Sets the parameterized string hour reference to the member variable m\_hour. |
| * ***void SetMinute(const string &minute);***   Used to set a month reference from the Time object.   * Sets the parameterized string minute reference to the member variable m\_day. |
| * ***void SetYear(const unsigned year);***   Used to set a year from the Date object.   * Sets the parameterized unsigned year to the member variable m\_year. |
| * ***ostream &operator << (ostream &outputStream, const Time &T);***   returns the output stream user defined Time object.   * Gets the hour, minute from and return the output stream. |
| * ***istream &operator << (istream &inputStream, const Time &T);***   returns the input stream user defined Time object.   * Sets the string hour, and minute from CSV and return the input stream. |

|  |
| --- |
| WindLogType |
| * ***WindLogType();***   Used for constructing a WindLogType object which sets all member variables to initial variable as 0.   * Sets member variable m\_windspeed to initial value float datatype of 0. * Sets member variable m\_airTemperature to initial value float datatype of 0. * Sets member variable m\_solarRadiation to initial value float datatype of 0. |
| * ***WindLogType(const Date &date, const Time &time, const float windspeed, const float airTemperature, const float solarRadiation);***   Used for constructing a WindLogType object that uses parameterized reference date object, time object, float windspeed, float airTemperature, and float solar radiation.   * Sets the reference date object to member variable m\_date. * Set the reference time object to the member variable m\_time. * Set a windspeed variable to the member variable m\_windspeed. * Set a float windspeed to the member variable m\_airTemperature. * Set a float solar radiation value to the member variable m\_solarRadiation. |
| * ***const Date GetDate() const;***   Used to get a date from the Date object.   * Returns the member variable Date object m\_date. |
| * ***const Date GetTime() const;***   Used to get a time from the Time object.   * Returns the member variable Time object m\_time. |
| * ***const float GetWindSpeed() const;***   Used to get a windspeed value from the WindLogType object.   * Returns the member variable m\_windspeed. |
| * ***const float GetAirTemperature() const;***   Used to get the air temperature value from the WindLogType object.   * Returns the member variable m\_airTemperature. |
| * ***const float GetSolarRadiation() const;***   Used to get the solar radiation value from the WindLogType object.   * Returns the member variable m\_solarRadiation. |
| * ***void SetDate(const Date &date) const;***   Used to set the date object value for the WindLogType object.   * Sets the parameterized date object to member variable m\_date. |
| * ***void SetTime(const Time &time) const;***   Used to set the time object value for the WindLogType object.   * Sets the parameterized time object to member variable m\_time. |
| * ***void SetWindSpeed(const float windspeed);***   Used to set the windspeed value for the WindLogType object.   * Sets the parameterized windspeed value to member variable m\_windspeed. |
| * ***void SetAirTemperature(const float airTemperature);***   Used to set the air temperature value for the WindLogType object.   * Sets the parameterized air temperature value to member variable m\_airTemperature. |
| * ***void SetSolarRadiation(const float solarRadiation);***   Used to set the solar radiation value for the WindLogType object.   * Sets the parameterized solar radiation value to member variable m\_solarRadiation. |
| * ***ostream &operator << (ostream &outputStream, const WindLogType &W);***   returns the output stream user defined WindLogType object.  Gets the date object, time object, windspeed, solar radiation, air temperature and return to the output stream. |
| * ***istream &operator << (istream &inputStream, const WindLogType &W);***   returns the input stream user defined WindLogType object.   * Sets the date object, time object, windspeed, solar radiation, air temperature from CSV and return the input stream. |

|  |
| --- |
| Menu |
| * ***void RunMenu(Vector<WindLogType>& windLogVector)***   This function holds the selection of choice from the user.   * Display the system allocated capacity. * Display the used spaces of the allocated dynamic array. * Do while loop the menu options. * Get the user option / choice. * A switch case in holding the option function as:   + OptionOne 🡪 display to console the average windspeed and air temperature on user specific month and year.   + OptionTwo 🡪 display the average speed and air temperature for each month of a specific user input year.   + OptionThree 🡪 display the solar radiation of each month of a specific user input year.   + OptionFour 🡪 display the average air temperature, total solar radiation for each user input specific year.   + OptionFive 🡪 exits the program by using break. |
| * ***void OptionOne(Vector<WindLogType>& windLogVector)***   This function holds the function in executing OptionTwo in the RunMenu function.   * Takes the user’s input specific month. * Check if user input of month is between 1 to 12, return if it’s false. * Takes the user’s input specific year. * Check if user input of year is between 2014 to 2016, return if it’s false. * Do a for loop in the vector in getting the user’s input of month to be equals to the vector’s month, and user’s input of year to be equals to vector’s year, and incurrment count +1 in every search success that meets the condition. * If the count is greater than 0 (which means month and year) were both found in the vector:   + Convert and calculate the average windspeed sum on the following (((sum / count) \*3.6) \* 10) / 10.   + Convert and calculate the average air temperature as: ((sum / count) \*10) / 10.   + Display to the console as *“month in string format year: the average windspeed km/h, average air temperature degrees C”* * Else   + Display “No Data” |
| * ***void OptionTwo(Vector<WindLogType>& windLogVector)***   This function holds the function in executing OptionTwo in the RunMenu function.   * Takes the user’s input specific year. * Check if user input of year is between 2014 to 2016, return if it’s false. * Initialize 3 array in the stack to hold:   + Creates a monthly sum windspeed array of 12 slots that holds initial value for each as 0.0.   + Creates a monthly sum temperature array of 12 slots that holds initial value for each as 0.0.   + Creates a monthly count array of 12 slots that holds initial value for each as 0. * Do a for loop in the vector in getting the user’s input of year to be equals to the vector’s year,   + - Get the month of the date -1,     - Sum the windspeed accordance to the month,     - Sum the air temperature as accordance to the month,     - and incurrment month count +1 in every search success that meets the condition. * Do a for loop of condition lesser than 12, increment month++   + If monthly count is greater than 0 (which means that there is a count from the user year input)     - Set the monthly windspeed sum array [month] to an initialized variable.       * Calculate and convert the average windspeed sum as: (((sum / count) \*3.6) \* 10) / 10.     - Set the monthly air temperature sum array [month] to an initialized variable.       * Calculate and convert the average air temperature sum as: ((sum / count) \*10) / 10.     - Set the monthly sum of sum [count] to an initialized variable.     - Display to console with the format*: “month in string format: average windspeed km/h, average air temperature degree C”.* * Else   + *Display month in string format: “No Data”* |
| * ***void OptionThree(Vector<WindLogType>& windLogVector)***   This function holds the function in executing OptionTwo in the RunMenu function.   * Takes the user’s input specific year. * Check if user input of year is between 2014 to 2016, return if it’s false. * Initialize 2 array in the stack to hold:   + Creates a monthly sum of solar radiation array of 12 slots that holds initial value for each as 0.0   + Creates a monthly count array of 12 slots that holds initial value for each as 0. * Do a for loop in the vector in getting the user’s input of year to be equals to the vector’s year,   + - Get the month of the date -1,     - Sum the solar radiation accordance to the month,     - and incurrment month count +1 in every search success that meets the condition. * Do a for loop of condition lesser than 12, increment month++   + If monthly count is greater than 0 (which means that there is a count from the user year input)     - Display to console with the format*: “month in string format: total sum of solar radiation as per month kWh/m2”.*       * Calculate and convert the total solar radiation sum as: ((sum / 6000) \* 10) / 10. * Else   + *Display month in string format: “No Data”* |
| * ***void OptionFour(Vector<WindLogType>& windLogVector)***   This function holds the function in executing OptionTwo in the RunMenu function.   * Takes the user’s input specific year. * Check if user input of year is between 2014 to 2016, return if it’s false. * Initialize a filename as WindTempSolar.csv file for outputting the data to. * Check if file exist:   + If Not, display failed to open the file for writing, and return. * Else   + Out file stream the header with month, average wind speed, average air temperature, and total solar radiation. * Initialize 4 array in the stack to hold:   + Creates a monthly sum of solar radiation array of 12 slots that holds initial value for each as 0.0.   + Creates a monthly sum windspeed array of 12 slots that holds initial value for each as 0.0.   + Creates a monthly sum temperature array of 12 slots that holds initial value for each as 0.0.   + Creates a monthly count array of 12 slots that holds initial value for each as 0. * Do a for loop in the vector in getting the user’s input of year to be equals to the vector’s year,   + - Get the solar radiation from the vector and store it in a variable.     - Get the windspeed from the vector and store it in a variable.     - Get the air temperature from the vector and store it in a variable.     - Get the month and store in a variable.     - Set the windspeed to the windspeed array and sum it.     - Set the air temperature to the aur temperature array and sum it.     - Set the total solar radiation to the array and sum it.     - Set the month count to the mount array. * Do a for loop of condition lesser than 12, increment month++   + Iterate through the months and write data to the CSV file     - * Calculate and convert the average windspeed sum as: (((sum / count) \*3.6) \* 10) / 10.       * Calculate and convert the average air temperature sum as: ((sum / count) \*10) / 10.       * Calculate and convert the total solar radiation sum as: ((sum / 6000) \* 10) / 10.   + Close the file.   + *Display a message that indicates that the file has been saved or written to.* |

|  |
| --- |
| Utils |
| * ***string GetMonthInStr(const unsigned &month) const***   This function returns a string format of a month.   * Initialise a string variable to hold the string format of a month in returning.   A switch case in holding a condition of a numeric month with the month in string format in every month as:   * + Case 1 🡪 January   + Case 2 🡪 February   + Case 3 🡪 March   + Case 4 🡪 April   + Case 5 🡪 May   + Case 6 🡪 June   + Case 7 🡪 July   + Case 8 🡪 August   + Case 9 🡪 September   + Case 10 🡪 October   + Case 11 🡪 November   + Case 12 🡪 December   + Default 🡪 empty string   Return month of the string format. |