# Data Dictionary

**Vector.h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| Vector<T> | Template |  | A template class that serves as a dynamic array |  |
| m\_pointer | T\* | - | pointer for the Vector |  |
| m\_size | unsigned int | - | The total number of elements stored |  |
| m\_capacity | unsigned int | - | The max number of elements that can be stored in Vector |  |
| Vector () |  | + | Constructor, sets m\_pointer to null, size and capacity to 0 |  |
| Vector (Vector& other) |  | + | Copy constructor, deep copies |  |
| ~Vector () |  | + | Destructor that will delete the pointers |  |
| Size () | unsigned int | + | Getter for m\_size |  |
| Capacity () | unsigned int | + | Getter for m\_capacity |  |
| operator=(Vector&) | void | + | Same as the copy constructor, deep copies | To allow user of the template to copy a Vector into another Vector. Takes in a Vector by reference as a parameter |
| Operator [] (const unsigned int& index) | const T& | + | Retrieves Vector element by index | To allow user to retrieve a vector's element value by a given index. A unsigned integer is required for the index |
| Resize (const unsigned int& newSize) | void | + | Changes the m\_size of the Vector to a given int |  |
| Erase (const unsigned int& index) | void | + | Removes a Vector element of a given index and move all elements on the right 1 step to the left |  |
| Insert (const unsigned int& position, const T& value) | void | + | Inserts an element into a specified index and move all elements from that index position 1 step right |  |
| Clear () | void | + | Set m\_size to 0 |  |
| Reserve (unsigned int newCapacity) | void | - | Increase the m\_capasity of Vector by 2 times if capacity is maxed |  |
| MultiplyBy2(const unsigned int& size) | unsigned int | - | Multiply an int by 2 |  |
| DeepCopy (T\* first, T\* last, T\* result) | T\* | - | Deep copies a Vector into result given the first element and the last element memory position |  |

**Date.h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| Date | class |  | A class for storing date | Each object is used to hold the date information after splitting them into the date components |
| m\_day | int | - | Stores day of the date |  |
| m\_month | int | - | Stores the month of the date |  |
| m\_year | int | - | Stores the year of the date |  |
| Date () |  | + | Constructor, sets m\_day, m\_month, m\_year to 0 |  |
| Date (unsigned int d, unsigned int m, unsigned int y) |  | + | Constructor, sets m\_day, m\_month, m\_year to what was given |  |
| ~Date () |  | + | Destructor, will not do anything as no pointers are used |  |
| GetDay () | int | + | Getter for m\_day |  |
| GetMonth () | int | + | Getter for m\_month |  |
| GetYear () | int | + | Getter for m\_year |  |
| SetDay (unsigned int d) | void | + | Setter for m\_day |  |
| SetMonth (unsigned int m) | void | + | Setter for m\_month |  |
| SetYear (unsigned int y) | void | + | Setter for m\_year |  |
| DisplayFullDate () | string | + | Returns a string containing the current date stored in DD/MM/YYYY format |  |
| operator<< (ostream&, const Date&) | Output stream | + | Overloaded insertion operator for output, allowing the Date object to be output to an output stream. |  |
| operator>> (istream&, Date&) | Input stream | + | Overloaded extraction operator for input, allowing the Date object to be input from an input stream. |  |

**Time.h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| Time | class |  | A class for storing time data | Each object will store a time in hour and minute |
| m\_hour | int | - | Stores hour of time in 24-hour time |  |
| m\_minute | int | - | Stores minute of time |  |
| Time () |  | + | Constructor sets m\_hour and m\_minute to -1 |  |
| Time (unsigned int h, unsigned int m) |  | + | Constructor sets m\_hour and m\_minute to given value |  |
| ~Time () |  | + | Destructor does nothing as no pointers are used |  |
| GetHour () | int | + | Getter for m\_hour |  |
| GetMinute () | int | + | Getter for m\_minute |  |
| SetHour (unsigned int h) | void | + | Setter for m\_hour |  |
| SetMinute (unsigned int m) | void | + | Setter for m\_hour |  |
| Display24HTime () | string | + | Returns a string containing the current time stored in object in HH:MM format |  |
| operator<< (ostream&, const Date&) | Output stream | + | Overloaded insertion operator for output, allowing the Date object to be output to an output stream. |  |
| operator>> (istream&, Date&) | Input stream | + | Overloaded extraction operator for input, allowing the Date object to be input from an input stream. |  |

**Weather.h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| Weather | Class |  | Object comprising of Date, Time, WindSpeed (float), SolarRadiation (float), AmbientAirTemperature (float). | Each object is used to hold a row of data from log data file. Holds the data as-is so that recreating the data is possible and there will be no loss of information |
| m\_date | Date | - | The date of a Log record |  |
| m\_time | Time | - | The time of Log record |  |
| m\_windSped | float | - | The Wind speed of Log record |  |
| m\_solarRad | float | - | The Solar Radiation of Log record |  |
| m\_airTemp | float | - | Ambient Air Temperature of Log record |  |
| Weather () |  | + | Constructor |  |
| ~Weather () |  | + | Destructor |  |
| GetDate () | Date& | + | Getter for m\_date |  |
| GetTime () | Time& | + | Getter for m\_time |  |
| GetWindSpeed () | float | + | Getter for m\_windsped |  |
| GetSolarRadiation () | float | + | Getter for m\_solarRad |  |
| GetAirTemperature () | float | + | Getter for m\_airtemp |  |
| SetDate (Date& date) | void | + | Setter for m\_date |  |
| SetTime (Time& time) | void | + | Setter for m\_time |  |
| SetWindSpeed (float windSpeed) | void | + | Setter for m\_windsped |  |
| SetSolarRadiation (float solar) | void | + | Setter for m\_solarRad |  |
| SetAirTemp (float temp) | void | + | Setter for m\_airtemp |  |

**Validator.h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| Validator | helper |  | A helper class used for validating incoming data that are being stored |  |
| IsValidDate (const int& day, const int& month, const int& year) | bool | + | Checks if a given set of date is valid. Day is 1-31 only Month is 1-12 only Year is 1000-9999 only If year is a leap year, then February would be checked that day is 1-29 instead of 1-28 Returns true if valid and false if invalid |  |
| IsValidDate (const int& month, const int& year) | bool | + | Checks if a given set of month and year are valid Month is 1-12 only Year is 1000-9999 only Returns true if valid and false if invalid |  |
| IsValidMonth (const int& month) | bool | + | Checks if given int is a valid month. Month is 1-12 only |  |
| IsLeapYear (const int& year) | bool | + | Checks if a given year is a leap year Returns true if it is and false if not |  |
| IsValid24HourTime (const int& hour, const int& minute) | bool | + | Checks if a given time is valid. Hour 0-24 only Time 0-59 only Return true if valid and false if not |  |
| IsNumber (const string& input) | bool | + | Checks if a given string is a valid number A number is valid if it contains digits and at most 1 decimal point The decimal point must be after at least 1 digit Returns true if valid and false if not |  |
| IsInteger (const string& input) | bool | + | Checks if a given string is a valid integer A integer is valid if it contains only digits Returns true if valid and false if not |  |

**Logic.h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| Logic | helper |  | A helper class used for holding functions relating to handling of logic | Class is used for things such as calculation, conversion, loading of data file and creating and writing of output file. This class will not deal with input and output to terminal |
| LoadDateFileCSV (ifstream& inputFile, Vector<Weather>& WindLog) | void | + | Takes in the file and a Vector It will loop through the file line by line and get only the date, time, wind speed, solar radiation, and air temperature data Each line of data will be stored in a weather object All data will be stored in Vector |  |
| DisplayMenu () | void | + | Displays the menu |  |
| RemoveLeadingZeros (string input) | string | + | Removes leading 0s from a given string |  |
| AverageWindSpeedMonth (const Vector<Weather>& array, const int month, const int year) | float | + | Search and calculates the average wind speed of a month when given the month and year to search for |  |
| AverageAirTemperatureMonth (const Vector<Weather>& array, const int month, const int year) | float | + | Search and calculates the average air temperature of a month when given the month and year to search for |  |
| SumSolarRadiationMonth (const Vector<Weather>& array, const int month, const int year) | float | + | Search and calculates the total solar radiation of a month when given the month and year to search for |  |
| IntMonthToString (const int& month) | string | + | Converts a given integer month to string month | For converting stored int month (1-12) to string (January-December) for output |
| ConvertWindMStoKMH (const float& windSpeed) | float | + | Multiplies a given float by 3.6 | This will help convert stored windSpeed which is m/s to km/h |
| ConvertSolarRadiationWMtoKWH (const float& radiation) | float | + | Divides a given float by 6 and 1000 | This will help convert stored solarRadiation which is in W/10mins to kWH |
| StandardDeviationWindSpeedMonth (const Vector<Weather>& array, const int month, const int year) | float | + | Calculate the standard deviation of wind speed per month. |  |
| StandardDeviationAirTemperatureMonth(const Vector<Weather>& array, const int month, const int year) | float | + | Calculate the standard deviation of ambient air temperature per month. |  |
| WriteAppendFile (const string& output) | void | + | Appends a given string to "WindTempSolar.csv" |  |

# Algorithm

## High Level

1. Get dataSourceFile.
2. Read the first line of dataSourceFile into inputFileName.
3. Set inputFile to open a file named "data/" concatenated with inputFileName.
4. Call LoadDataFileCSV on Logic with inputFile and WindLog as parameters.
5. Display menu
6. If option 1 is chosen
   1. Ask user for month and year input.
   2. Search Vector for WindLogType with same month and year in date
   3. Sum all the wind speed and count the number of WindLogType that matches.
   4. Divide wind speed by count to get average.
   5. Calculate the standard deviation of wind speed.
   6. Convert wind speed from m/s to km/h.
   7. Print result.
7. If option 2 is chosen
   1. Ask user for year input.
   2. Search Vector for WindLogType with same year and month is 1 in date.
   3. Sum all ambient air temperature and count the number of WindLogType that matches.
   4. Divide air temperature by count to get average.
   5. Calculate the standard deviation of ambient air temperature.
   6. Prints result.
   7. Repeat step 7.2-7.6 for month 2-12
8. If option 3 is chosen
   1. Ask user for year input.
   2. Search vector for WindLogType with same year and month is 1 in date.
   3. Sum all solar radiation.
   4. Convert W/m^2 to kWh/m^2.
   5. Print result
   6. Repeat step 8.2 – 8.5 for month 2-12
9. If option 4 is chosen
   1. Ask user for year input.
   2. Search vector for WindLogType with same year and month is 1 in date.
   3. Get Average wind speed, standard deviation for wind speed, Average ambient air temperature, standard deviation for air temperature and total solar radiation (use functions from made for option1, option 2 and 3 for this)
   4. print to a file called “WindTempSolar.csv”.
   5. Repeat step 9.2 – 9.4 for month 2-12.
10. If option 5 is chosen
    1. Exit program.