**Data Dictionary:**

Vector Class:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Protection | Description |
| Vector | Class |  | Holds a vector without using stl |
| m\_vector | T\* (pointer) | - | Creates a vector |
| m\_size | int | - | Number of elements in the vector |
| m\_capacity | int | - | Maximum number of elements a vector can hold |
| Vector() | Constructor | + | Initialize empty vector |
| ~Vector() | Destructor | + | Frees an allocated memory |
| Insert(const T& data, int index) | bool | + | Insert elements at a specific index, shifting elements to the right |
| Add(const T& data) | void | + | Add elements |
| GetSize() | int | + | Return number of elements in the vector |
| GetCapacity() | int | + | Returns maximum number of elements vector can hold |
| operator[int index] | const T& | + | Overloads to allow read access |
| operator[int index] | T& | + | Overloads to allow modification |

Date Class:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Protection | Description |
| Date | Class |  | Holds data for date |
| m\_day | int | - | Day of the date |
| m\_month | int | - | Month of the date |
| m\_year | int | - | Year of the date |
| Date() | Constructor | + | Initialize |
| Date(int day, int month, int year) | Constructor | + | Holds the day, month and year of the date |
| GetDay() | int | + | Return the day of the date |
| SetDay(int day) | void | + | Set the day of the date |
| GetMonth() | int | + | Return the month of the date |
| SetMonth(int month) | void | + | Set the month of the date |
| GetYear() | int | + | Return the year of the date |
| SetYear(int year) | void | + | Set the year of the date |

Time Class:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Protection | Description |
| Time | Class |  | Holds data for time |
| m\_hour | int | - | Hour of the time |
| m\_minute | int | - | Minute of the time |
| m\_second | int | - | Second of the time |
| Time() | Constructor | + | Initialize |
| Time(int hour, int minute, int second) | Constructor | + | Holds the hour, minute and second of the time |
| GetHour() | int | + | Return the hour of the time |
| SetHour() | void | + | Set the hour of the time |
| GetMinute() | int | + | Return the minute of the time |
| SetMinute() | void | + | Set the minute of the time |
| GetSecond() | int | + | Return the second of the time |
| SetSecond() | void | + | Set the second of the time |

WeatherData Class:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Protection | Description |
| WeatherData | Class |  | Holds data for weather conditions |
| m\_speed | float | - | Wind speed |
| m\_airTemp | float | - | Ambient air temperature |
| m\_solarRad | float | - | Solar radiation |
| m\_date | Date | - | Date from date class |
| m\_time | Time | - | Time from time class |
| WeatherData() | Constructor | + | Initialize |
| GetSpeed() | float | + | Return the wind speed |
| SetSpeed(float speed) | void | + | Set the wind speed |
| GetAirTemp() | float | + | Return the ambient air temperature |
| SetAirTemp(float airTemp) | void | + | Set the ambient air temperature |
| GetSolarRad() | float | + | Return the solar radiation |
| SetSolarRad(float solarRad) | void | + | Set the solar radiation |
| GetDate(Date & date) | void | + | Return the date from date class |
| SetDate(const Date & date) | void | + | Set the date from date class |
| GetTime(Time & time) | void | + | Return the time from time class |
| SetTime(const Time & Time) | void | + | Set the time from time class |

LoadRecords Class:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Protection | Description |
| LoadRecords | Class |  | Handles data from the CSV file and to store them in a vector |
| LoadRecords(Vector <WeatherData> & dataRecords, const string & filename) | int | + | Loads data from the CSV file and store them in a vector |
| CSV\_DELIMITER | char | - | Delimiter used to separate columns in the CSV file (,) |
| DATE\_DELIMITER | char | - | Delimiter used to separate the date elements in MM/DD/YYYY (/) |
| TIME\_DELIMITER | char | - | Delimiter used to separate the time elements in HH:MM (:) |
| inFile | ifstream | - | Input file stream to read the CSV file |
| header | string | - | Variable to hold the CSV header |
| newHeader | stringstream | - | String stream to process the header row from the CSV file |
| column | string | - | Variable to store each column value in the header |
| columnidx | int | - | Index of the current column |
| dateCol | int | - | Column for date in CSV file |
| speedCol | int | - | Column for speed in CSV file |
| srCol | int | - | Column for solar radiation in CSV file |
| tCol | int | - | Column for temperature in CSV file |
| line | string | - | Variable to hold each data line |
| tempDateStr | string | - | String to store the date as MM/DD/YYYY |
| tempTimeStr | string | - | String to store the time as HH:MM |
| tempStr | string | - | String to extract individual data values from the CSV line |
| windSpeed | float | - | Wind speed in meters per second, later converted to km/h |
| airTemp | float | - | Air temperature in degrees Celsius |
| solarRad | float | - | Solar radiation in kWh/m² |
| ss | stringstream | - | String stream to process each data line |
| ssDate | stringstream | - | String stream to process the date string (MM/DD/YYYY) |
| ssTime | stringstream | - | String stream to process the time string (HH:MM) |

DataCalculations Class:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Protection | Description |
| DataCalculations | Class |  | Holds all the data calculations |
| averageSpeed | float | - | Average wind speed |
| averageAirTemp | float | - | Average ambient air temperature |
| DataCalculations() | Constructor | - | Initialize |
| CalculateAvgSpeed(Vector <WeatherData> & dataRecords) | float | + | Calculate the average wind speed |
| CalculateSpeedSD(Vector <WeatherData> & dataRecords, float averageSpeed) | float | + | Calculate the standard deviation for wind speed |
| CalculateAvgAirTemp(Vector <WeatherData> & dataRecords) | float | + | Calculate the average ambient air temperature |
| CalculateAirTempSD(Vector <WeatherData> & dataRecords, float averageAirTemp) | float | + | Calculate the standard deviation for ambient air temperature |
| CalculateSolarRad(Vector <WeatherData> & dataRecords) | float | + | Calculate the total solar radiation |

Main Class:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Protection | Description |
| main | int |  | Contains data and menu |
| dataRecords | Vector <WeatherData> | - | Stores data records loaded from a CSV file |
| dataCal | dataCalculations | - | Handles the calculation |
| option | string | - | Stores the user’s menu option input |
| averageSpeed | bool | - | Stores the calculated average wind speed |
| averageAirTemp | void | - | Stores the calculated average ambient air temperature |
| speedSD | int | - | Stores the calculated standard deviation for wind speed |
| airTempSD | int | - | Stores the calculated standard deviation for ambient air temperature |
| solarRadiation | const T& | - | Stores the calculated solar radiation |
| option == "1" | if statement | - | Checks if the option is "1" and prints the weather data records |
| option == "2" | if statement | - | Checks if the option is "2" and calculates average wind speed |
| option == "3" | if statement | - | Checks if the option is "3" and calculates average ambient air temperature |
| option == "4" | if statement | - | Checks if the option is "4" and calculates standard deviation for speed and air ambient temperature |
| option == "5" | if statement | - | Checks if the option is "5" and calculates total solar radiation |
| option == "6" | if statement | - | Checks if the option is "6" and exit the program |
| Invalid Input | else statement | - | Prints a message when an invalid menu option is entered |

**Algorithm:**

**Date.cpp:**

CONSTRUCTOR Date():

INITIALIZE m\_day = 0

INITIALIZE m\_month = 0

INITIALIZE m\_year = 0

END CONSTRUCTOR

CONSTRUCTOR Date(int day, int month, int year):

INITIALIZE m\_day = day

INITIALIZE m\_month = month

INITIALIZE m\_year = year

END CONSTRUTOR

FUNTION GetDay() : int

return m\_day

END FUNCTION

FUNCTION SetDay(int day):

set m\_day = day

END FUNCTION

FUNCTION GetMonth() : int

return m\_month

END FUNCTION

FUNCTION SetMonth(int month):

set m\_month = month

END FUNCTION

FUNCTION GetYear() : int

return m\_year

END FUNCTION

FUNCTION SetYear(int year):

set m\_year = year

END FUNCTION

OVERLOAD Operator << (ostream & os, const Date & date):

Print "Date: " followed by m\_day, m\_month, m\_year to output stream (formatted as "day/month/year")

return output stream

END OPERATOR <<

OVERLOAD Operator >> (istream & input, Date & date):

read day from input stream

set m\_day to the input day

read month from input stream

set m\_month to the input month

read year from input stream

set m\_year to the input year

return input stream

END OPERATOR >>

**Time.cpp:**

CONSTRUCTOR Time():

INITIALIZE m\_hour = 0

INITIALIZE m\_minute = 0

INITIALIZE m\_second = 0

END CONSTRUCTOR

CONSTRUCTOR Time(int hour, int minute, int second):

INITIALIZE m\_hour = hour

INITIALIZE m\_minute = minute

INITIALIZE m\_second = second

END CONSTRUTOR

FUNTION GetHour() : int

return m\_hour

END FUNCTION

FUNCTION SetHour (int hour):

set m\_hour = hour

END FUNCTION

FUNCTION GetMinute() : int

return m\_minute

END FUNCTION

FUNCTION SetMinute(int minute):

set m\_month = month

END FUNCTION

FUNCTION GetSecond() : int

return m\_second

END FUNCTION

FUNCTION SetSecond(int second):

set m\_second = second

END FUNCTION

OVERLOAD Operator << (ostream & os, const Time & time):

Print "Time: " followed by m\_hour, m\_minute, m\_second to output stream (formatted as "HH:MM:S")

return output stream

END OPERATOR <<

OVERLOAD Operator >> (istream & input, Time & time):

read hour from input stream

set m\_hour to the input hour

read minute from input stream

set m\_minute to the input minute

read second from input stream

set m\_second to the input second

return input stream

END OPERATOR >>

**WeatherData.cpp:**

CONSTRUCTOR WeatherData():

INITIALIZE m\_speed = 0

INITIALIZE m\_airTemp = 0

INITIALIZE m\_solarRad = 0

END CONSTRUCTOR

FUNTION GetSpeed() : float

return m\_speed

END FUNCTION

FUNCTION SetSpeed(float speed):

set m\_speed = speed

END FUNCTION

FUNCTION GetAirTemp() : float

return m\_airTemp

END FUNCTION

FUNCTION SetAirTemp(float airTemp):

set m\_airTemp = airTemp

END FUNCTION

FUNCTION GetSolarRad() : float

return m\_solarRad

END FUNCTION

FUNCTION SetSolarRad(float solarRad):

set m\_solarRad = solarRad

END FUNCTION

FUNCTION GetDate(Date & date) :

return date = m\_date

END FUNCTION

FUNCTION SetDate(const Date & date):

set m\_date = date

END FUNCTION

FUNCTION GetTime(Time & time) :

return time = m\_time

END FUNCTION

FUNCTION SetTime(Time & time):

set m\_time = time

END FUNCTION

**LoadRecords.cpp:**

FUNCTION LoadRecords(Vector <WeatherData> & dataRecords, const string & filename) : int

INITIALIZE int count = 0

Open file for reading

DEFINE CSV\_DELIMITER as ','

DEFINE DATE\_DELIMITER as '/'

DEFINE TIME\_DELIMITER as ':'

Read the header line from file

Process header to find indexes of WAST, S, SR, T

INITIALIZE string column

INITIALIZE int columnidx = 0

INITIALIZE int dateCol = -1

INITIALIZE int speedCol = -1

INITIALIZE int srCol = -1

INITIALIZE int tCol = -1

WHILE each column in header (split by CSV\_DELIMETER)

IF column is "WAST"

set dateCol to columnidx

END IF

IF column is "S"

set speedCol to columnidx

END IF

IF column is "SR"

set srCol to columnidx

END IF

IF column is "T"

set tCol to columnidx

END IF

Increment columnidx

END WHILE

INITIALIZE string line

INITIALIZE string tempDateStr

INITIALIZE string tempTimeStr

INITIALIZE string tempStr

INITIALIZE float windSpeed

INITIALIZE float airTemp

INITIALIZE float solarRad

WHILE line read from file

Read line and split by CSV\_DELIMITER

Extract tempDateStr and tempTimeStr

Convert tempDateStr to (day, month, year)

Convert tempTimeStr to (hour, minute)

FOR i = 1 to columnidx - 1

Read next value from CSV

IF i == speedCol

convert to windSpeed (m/s to km/h)

END IF

IF i == tCol

store as airTemp

END IF

IF i == srCol

store as solarRad

END IF

END FOR

CREATE WeatherData record

Set record date

Set record time

Set record speed

Set record airTemp

Set record solarRad

Add record to dataRecords

Increment count

END WHILE

RETURN count

END FUNCTION

**DataCalculations.cpp:**

CONSTRUCTOR DataCalculations()

Initialize averageSpeed = 0

Initialize averageAirTemp = 0

END CONSTRUCTOR

FUNCTION CalculateAvgSpeed(Vector <WeatherData> & dataRecords) : float

INITIALIZE float sum = 0

FOR i = 0 to dataRecords.GetSize() - 1

ADD dataRecords[i].GetSpeed() to sum

END FOR

RETURN sum / dataRecords.GetSize()

END FUNCTION

FUNCTION CalculateSpeedSD(Vector <WeatherData> & dataRecords, float averageSpeed) : float

INITIALIZE sum = 0

FOR i = 0 to dataRecords.GetSize() - 1

sum = COMPUTE (dataRecords[i].GetSpeed() – averageSpeed) \* (dataRecords[i].GetSpeed() - averageSpeed)

END FOR

RETURN sqrt(sum / (dataRecords.GetSize() - 1))

END FUNCTION

FUNCTION CalculateAvgAirTemp(Vector <WeatherData> & dataRecords) : float

INITIALIZE sum = 0

FOR i = 0 to dataRecords.GetSize() - 1

ADD dataRecords[i].GetAirTemp() to sum

END FOR

RETURN sum / dataRecords.GetSize()

END FUNCTION

FUNCTION CalculateAirTempSD(Vector <WeatherData> & dataRecords, float averageAirTemp) : float

INITIALIZE sum = 0

FOR i = 0 to dataRecords.GetSize() - 1

sum = COMPUTE (dataRecords[i].GetSpeed() – averageAirTemp) \* (dataRecords[i].GetSpeed() - averageAirTemp)

END FOR

RETURN sqrt(sum / (dataRecords.GetSize() - 1))

END FUNCTION

FUNCTION CalculateSolarRad(Vector <WeatherData> & dataRecords) : float

INTIALIZE sum = 0

FOR i = 0 to dataRecords.GetSize() - 1

ADD dataRecords[i].GetSolarRad() to sum

END FOR

RETURN sum

END FUNCTION

Main.cpp:

FUNCTION main() : int

CREATE Vector <WeatherData> dataRecords

LoadRecords (dataRecords, FileName)

CREATE DataCalculations dataCal

INITIALIZE string object = " "

DO

PRINT Menu Options

PRINT 1.Print Records

PRINT 2.Calculate Average Wind Speed

PRINT 3.Calculate Average Ambient Air Temperature

PRINT 4.Calculate Standard Deviation

PRINT 5.Calculate Solar Radiation

PRINT 6.Exit

PRINT Choose Your Option:

STORE option

INITIALIZE float averageSpeed

INITIALIZE float averageAirTemp

INITIALIZE float speedSD

INITIALIZE float airTempSD

INITIALIZE float solarRadiation

IF option == 1

DISPLAY dataRecords.GetSize()

FOR i = 0 to dataRecords -1

Get and display Date

Get and display Time

Get and display Speed (km/h)

Get and display Air Temperature (°C)

Get and display Solar Radiation (kWh/m²)

END FOR

END IF

ELSE IF option == 2

GET averageSpeed = dataCal.CalculateAvgSpeed(dataRecords);

DISPLAY Average Wind Speed(km/h)

END ELSE IF

ELSE IF option == 3

GET averageAirTemp = dataCal.CalculateAvgAirTemp(dataRecords);

DISPLAY Average Ambient Air Temperature (°C)

END ELSE IF

ELSE IF option == 4

GET speedSD = dataCal.CalculateSpeedSD(dataRecords, averageSpeed);

DISPLAY Speed Standard Deviation

GET airTempSD = dataCal.CalculateAirTempSD(dataRecords, averageAirTemp);

DISPLAY Ambient Air Temperature Standard Deviation

END ELSE IF

ELSE IF option == 5

GET solarRadiation = dataCal.CalculateSolarRad(dataRecords);

DISPLAY Total Solar Radiation: (kWh/m²)

END ELSE IF

ELSE IF option == 6

DISPLAY Exiting Program

END ELSE IF

ELSE

DISPLAY Invalid Input! Please enter number from 1 to 6.

END ELSE

END DO

WHILE option != 6

DISPLAY ~End Of Program~

return 0;

END FUNCTION

**Test Plan:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test | Description | Actual Test Data | Expected Output | Passed |
| 1 | Load CSV File | Records loaded successfully | Load File and show their data inside the file | Yes |
| 2 | Print all records | Select option 1 | Show all record size, date, time, wind speed, air temperature, solar radiation | Yes |
| 3 | Calculate average wind speed | Select option 2 | Shows the average wind speed | Yes |
| 4 | Calculate average air temperature | Select option 3 | Shows the average ambient air temperature | Yes |
| 5 | Calculate standard deviation | Select option 4 | Shows the standard deviation of wind speed and ambient air temperature | Yes |
| 6 | Calculate solar radiation | Select option 5 | Shows total solar radiation | Yes |
| 7 | Exit | Select option 6 | Exit program | Yes |
| 8 | Invalid input | Entering number outside of 1 to 6 | Shows that input is invalid | Yes |