Point2D.h and point2D.cpp

Text

Description automatically generated

Point2D class follows the UML diagram requirements of the class, with and addition of 3 friend functions that overload the operators. The insertion, less than and more than operators are overloaded so that my function template can use them when it is being called. Trying out the usage of global variables, filter, sortC and order so that all my files can access them. I am required to use the extern key word so that it does not have multiple initializations of the variable.

Text

Description automatically generated

For the insertion operator, using the #include <iomanip> it allows me to format my Ostream. setW allows me to set the width and reserve the 4 spaces for my values. Fixed and setprecision allows me to specify the number of dp the printed output displays.

For the less than operator, I start by checking what type of sort criteria is chosen. From there I have a string of if-else statements to do my comparing. As mentioned in the assignment, if the comparison for the first value is the same, we go and compare the x-value next, then the y-value and if required the z-value. In this case, if my initial x-value is the same, I compare the y-value.

The same is true for the greater than operator overloading. The rest of the code in this class is as stated in the requirements of the assignment. Default constructor, constructor and the get and set methods.

Point3D.h and Point3D.cpp

A screenshot of a computer

Description automatically generated with medium confidence

Similar to Point2D I only overloaded 3 operators. The rest of the functions are as per normal.

Text

Description automatically generated

For the operator overloading of Point3D there is one more variable to deal with. As we can see from the if-else statement. If x is the same, we compare y. if x and y are the same, we compare z. same goes, if y is the same, we compare x, and if y and x is the same we compare z. Length is the variable of the parent child so there is no need for this class to get a get method for the distFrOrigin. As it can be displayed through the parent getScalarValue() call.

Line2D.h Line2D.cpp

Text

Description automatically generated

Line2D contains the same 3 operator overloads. It required the #include “Point2D.h” as it uses that class type for some of the variables.

Text

Description automatically generated

Line2D has the method getScalarValue() which similar to Point2D, will be accessed by its child class, Line3D. The rest of the functions follow suit.

Line3D.h Line3D.cpp

Text

Description automatically generated

Line3D includes Line2D.h as it is its parents class, it also include Point3D.h as it uses that class type as its variable. Operator overloading follows the previous classes. The rest of the functions are as required.

Text

Description automatically generated

Length is the variable in parent class Line2D where it can be accessed by the child class. The method getScalarValue is also used by Line3D to output the length.

Functions.h Functions.cpp

Text

Description automatically generated

All the functions required by the program in the header file. All the inclusions of the headers also required by the program.

MainMenu()

Text

Description automatically generated

Mainmenu option is put into a do-while loop in the main so that the program will not terminate unless the user chooses to.

OptionOne

Text

Description automatically generated

For the readIn option, instead of using a function template with the operator == overloaded to compare duplicates, I compare and remove duplicates before I start processing the data. This is done by taking all the data in the file and putting them into an array. I then sort the array and use the keyword unique to remove duplicate data in the sorted array. After this is done I then process the sort and unique data through if-else statements.

Text

Description automatically generated

The if-else statements check for the object type, creates the object and stores them in their own vector.

OptionTwo

Text

Description automatically generated

A Simple switch case that sets the global variable to what the user has chosen and also resets the sortC to the correct “default” value

OptionThree

Text

Description automatically generated

Slightly different way of handling the user input. As the sort criteria given is based on what the filter criteria is. So we have an if statement for every filter criteria so that it can customize what it shows the user.

optionFour

Text

Description automatically generated

Option four is straight forward, only requires the user to choose from ascending or descending.

optionFive

Text

Description automatically generated

Through the selected user filter, sort and order criteria, optionFive will correctly display what the user wants through the use of if and else-if statements. Since the global variables are accessible by this function, it first checks if order is ASC or DSC as it determines which function template it will use as the comparator in the sort function. Next it couts the table header through the keywords from #include<iomanip> then through a for loop, with the insertion operator overloaded, we simply just cout << the object. This applies for all the other filter criteria

OptionSix

Text

Description automatically generated

OptionSix first takes in the user input for a file name, it then creates an fStream object with the filename and opens the file. The following methods are the same as in optionFive. However instead of cout we use file. This allows the ostream to output directly into the file