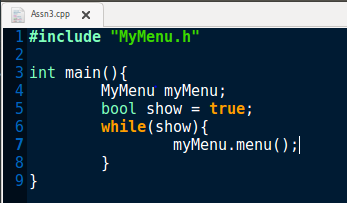
CSCI251 Advanced Programming

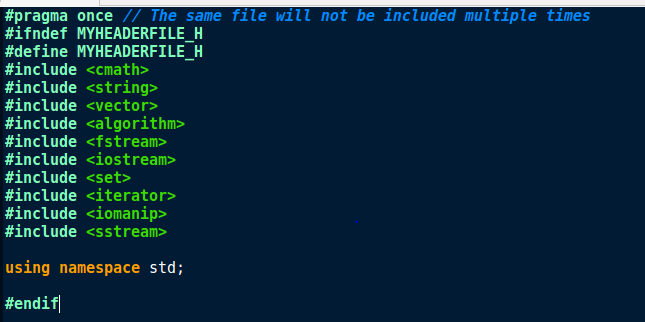
Report For Assignment 3

Main File (Assn3.cpp)



Assn3.cpp is the main file for whole program for this assignment. Inside the main function, I have call the MyMenu which is the class name and give it a name as myMenu for menu function to let system to run out the menu function.

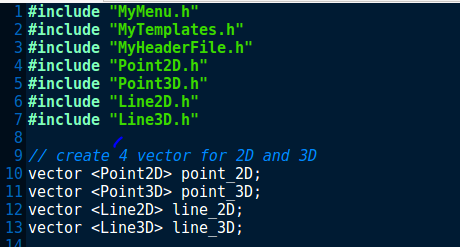
MyHeaderFile.h



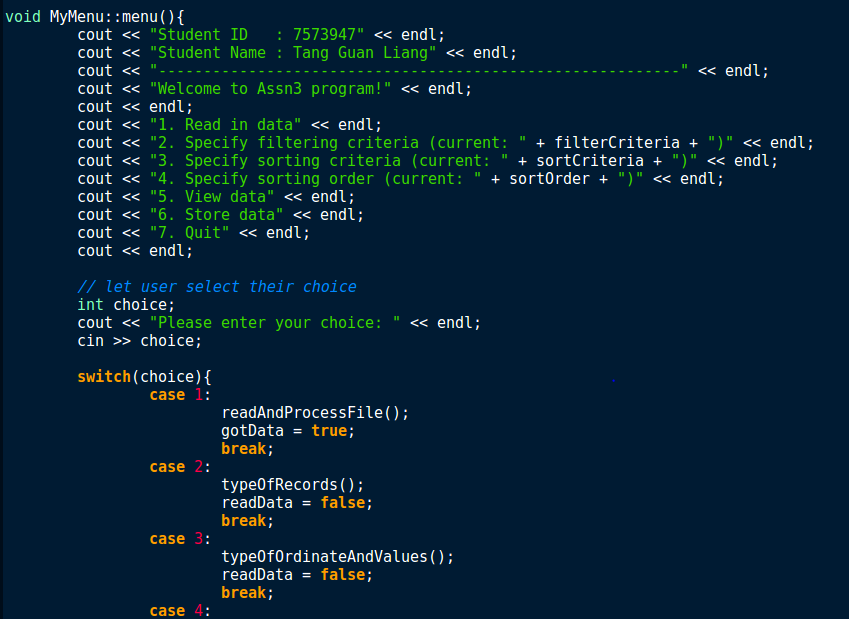
These are the preprocessor directive, include statement and header guards that I use in this assignment and this file. For pragma once, it ensures that the file will not be included multiple times in a same program. For the include statement, it tells the compiler which file from the C++ standard library to use in the program.

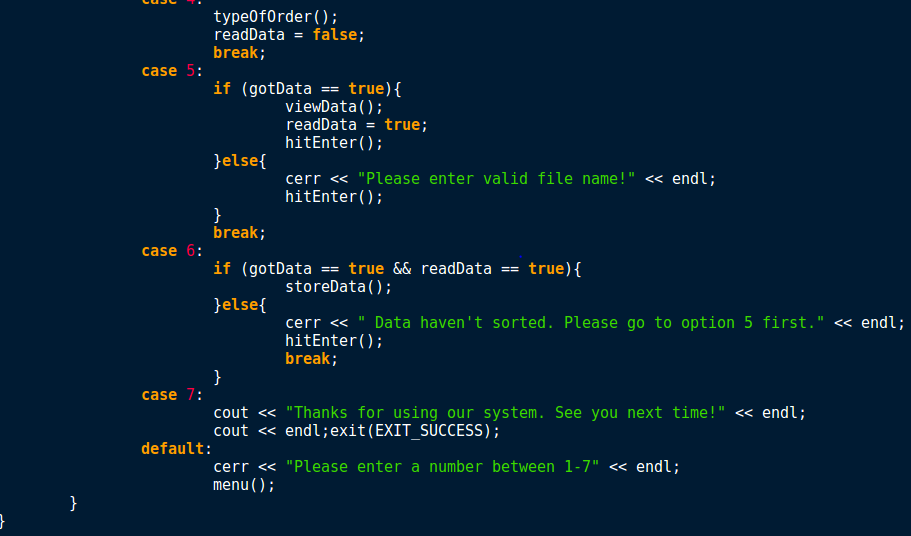
Menu function

MyMenu.cpp

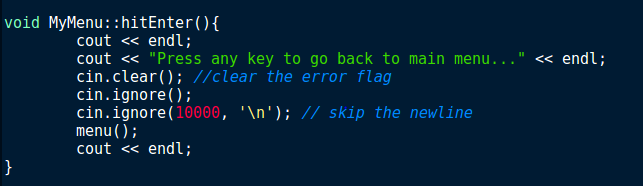


These are the include statement that use in this file. I also have created 4 vectors for Point2D, Point3D, Line2D and Line3D so that the data can be stored in their respective vectors.





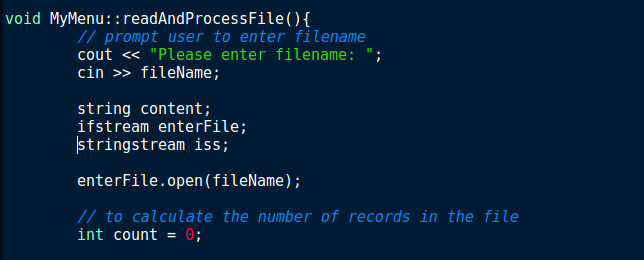
These are the menu part for this assignment. It shows the 7 options that user can choose and the functions that will execute in each option. Besides, it also has shown the error message for each option and in menu if user didn’t follow the instructions of this program. For the option 2 to 4, it also shows that the type of filtering criteria, sorting criteria and sorting order that user currently choose to deal with data. For the option 7, the compiler will print out the message and terminate the program.



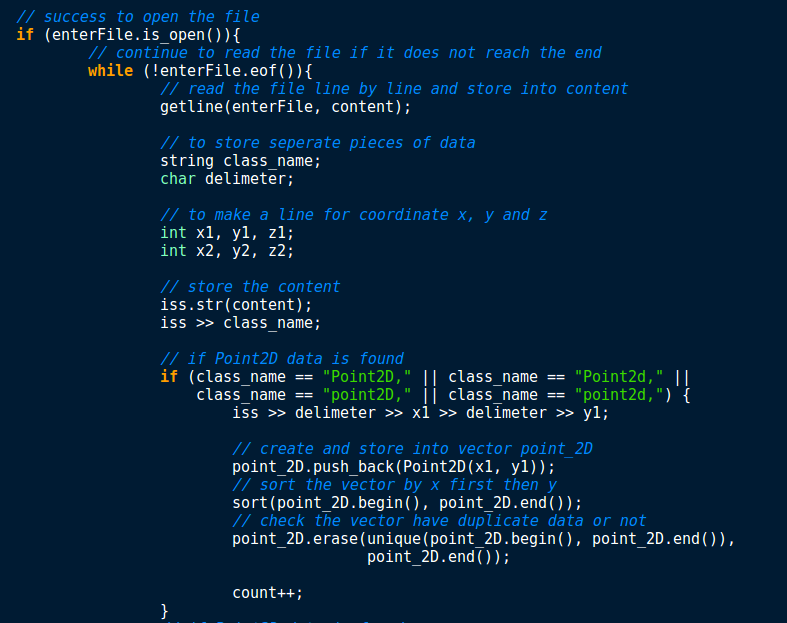
This is a function that can let user press any button to go back to main menu.

Functions for options

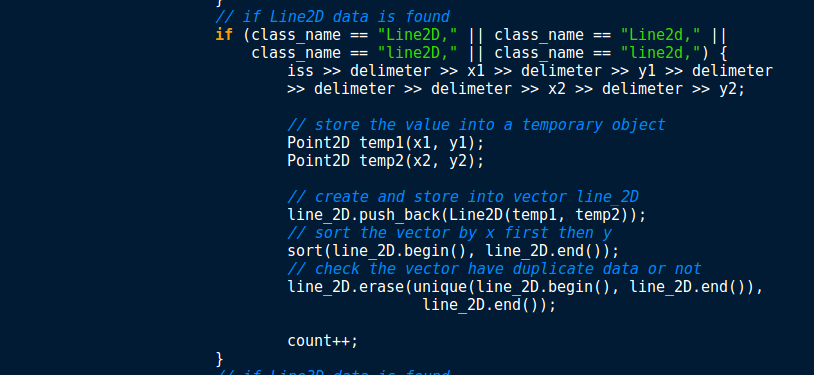
Option 1



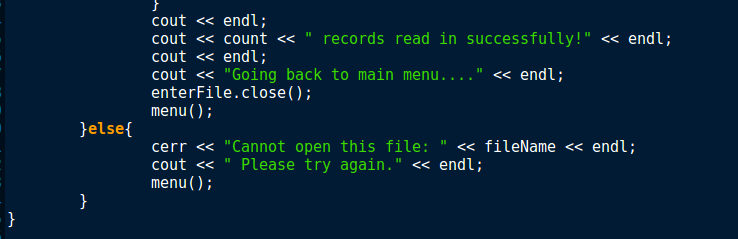
This is the function that will use for option 1. When user choose option 1, it will prompt a message to let user to enter the file name. When the program successfully opens the file, it will count the number of records in this file.



If the program success to open the file, the program will continue read the file line by line and will store the data into content until reach the end of file. For storing the data, the program will separate the data into pieces instead of just store whole data in one index. If Point2D data is found, it will create the symbols and store the data into vector for point 2D. Next, the program will sort the data that in vector by coordinate x first then coordinate y and check the vector have duplicate data or not. For Point3D also same as Point2D except for the store and sort part, it needs to add coordinate z and sort as coordinate x first then coordinate y finally coordinate z.

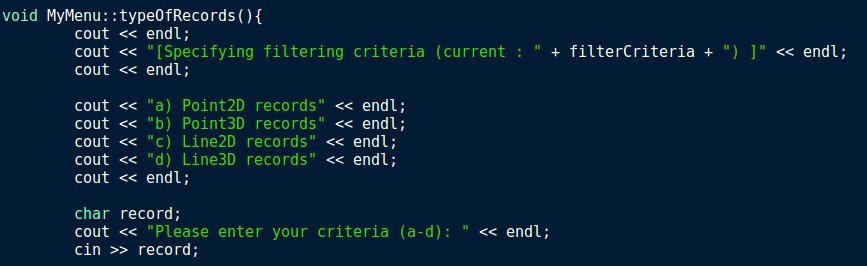


For Line2D and Line3D, it just has a bit difference with Point2D and Point3D. If Line2D and Line3D data is found, the program will store the value into a temporary object. After that, create a vector for Line2D and Line3D and store the temporary object into vector. The rest are same as Point2D and Point3D.

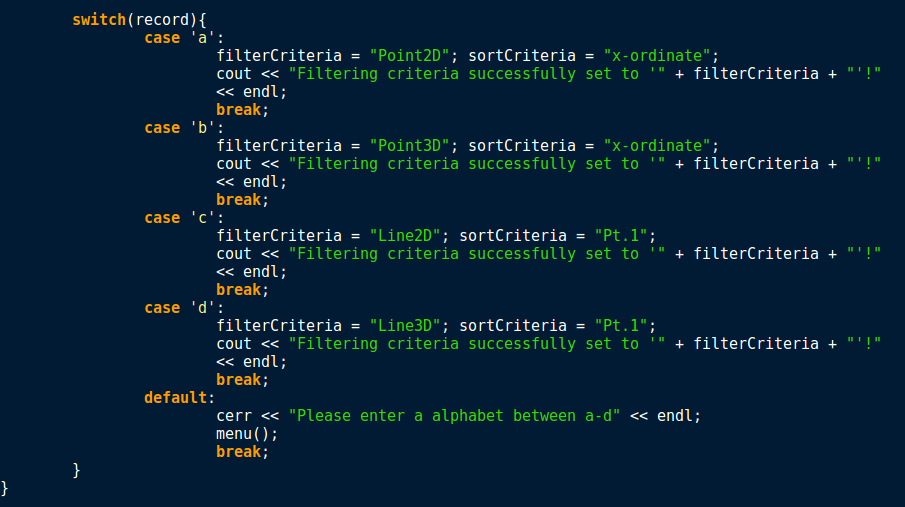


After finish storing each of the data into each of the vector, the program will the message that how many records read in successfully and go back to menu. However, if the program cannot open the file name that user enter at the beginning, the program will print out the error message and go back to menu.

Option 2

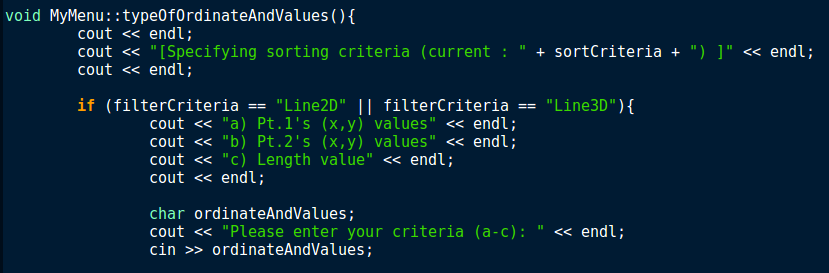


This is the function that will use for option 2. When user choose option 2, it will print out a message and show the default filtering criteria which is Point2D. Next, the program will print out four options to let user choose.

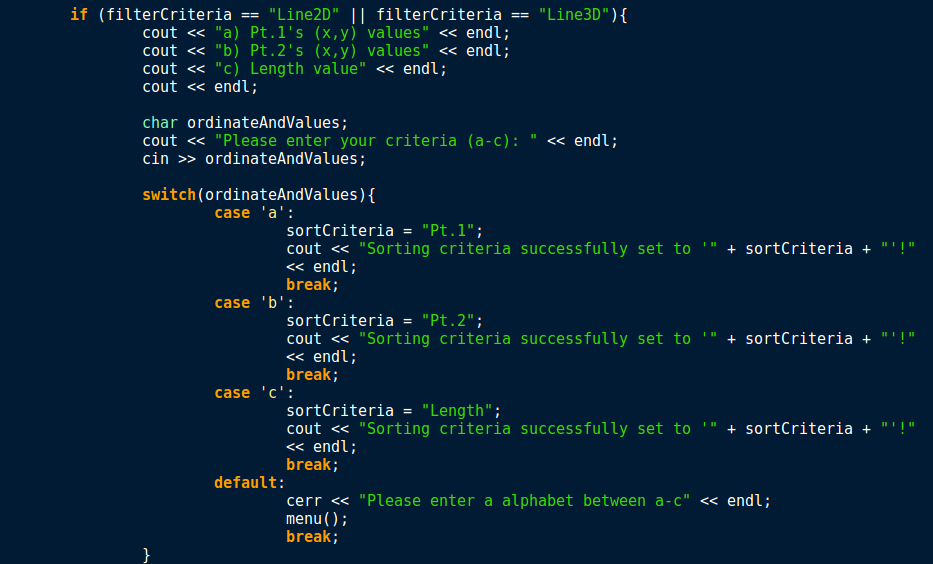


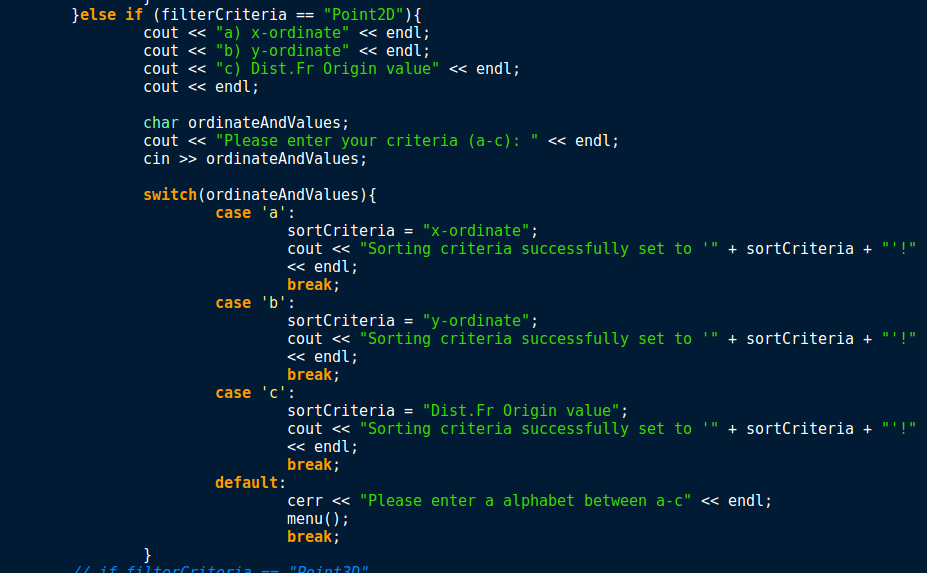
These are the conditions that after user choose the four options and the message that will print out. For example, the default filtering criteria is Point2D. If user choose option b which is Point3d records, the program will see the conditions for b, if the condition is all correct then the program will print out a message that filtering criteria successfully set to Point3D and the option 2 that in menu also will change to Point3D. However, user only can enter alphabet to choose the option. If user type in numbers or another alphabet which is not between a to d. The program will print out the error message and go back to menu.

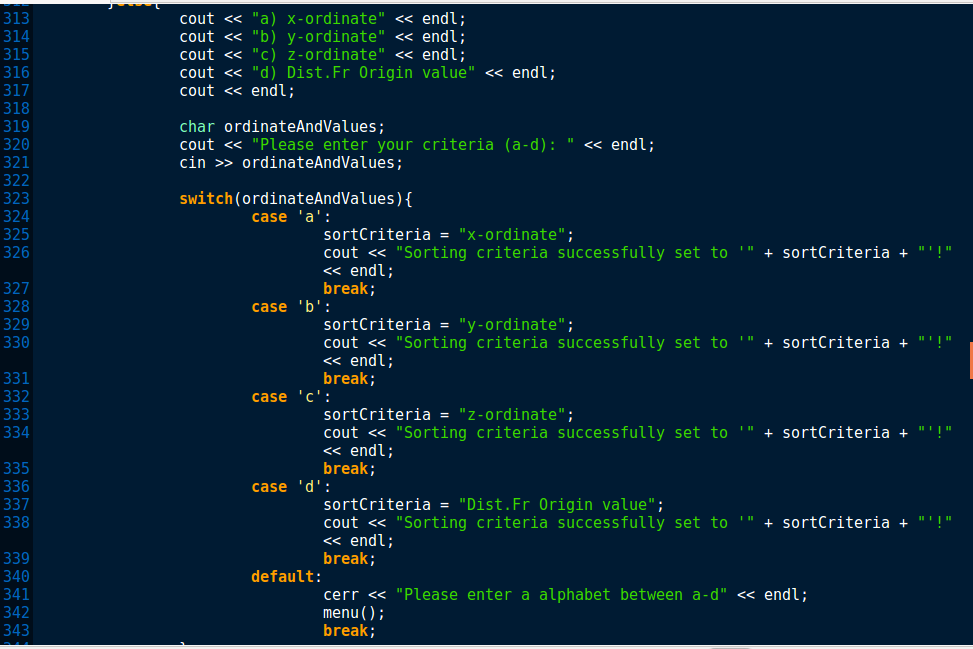
Option 3



This is the function that will use for option 3. When user choose option 3, it will print out a message and show the default sorting criteria which is x-ordinate.

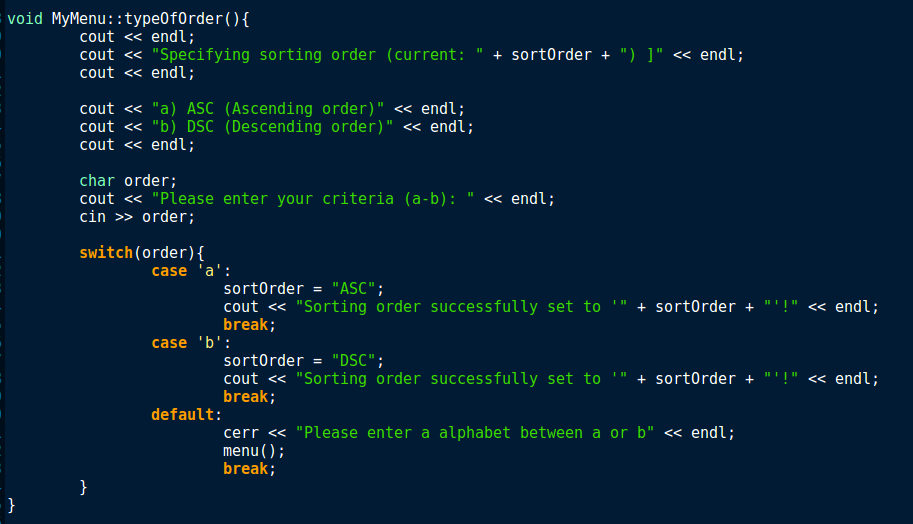






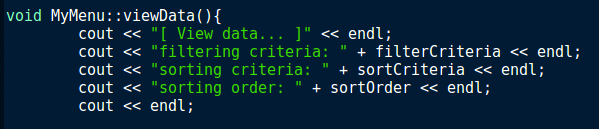
These are the conditions that will happen in option 3, the condition will be different according to the choice that user choose in option 2. For example, if user choose Point3D records in option 2, when user comes to option 3, the program will find the condition for Point3D and give user the sorting criteria options that only for Point3D.

Option 4

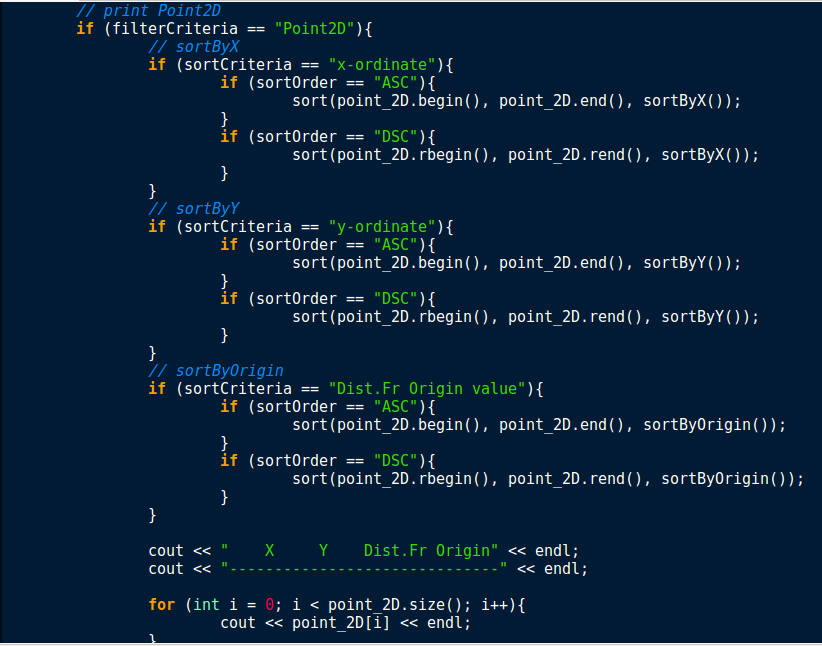


This is the function that will use for option 4. When user choose option 4, it will print out a message and show the default sorting order which is ascending order (ASC). Unlike option 3 which have the conditions based on what user have choose in option 2. This function is just showing two options for sorting order. One is ascending order (ASC), another one is descending order (DSC). This is just let user to choose what order that user want to present the data. It same as option 2 and 3 which if user type in a number or other alphabet such as not between a to b in this case, the program will print out the error message and go back to menu.

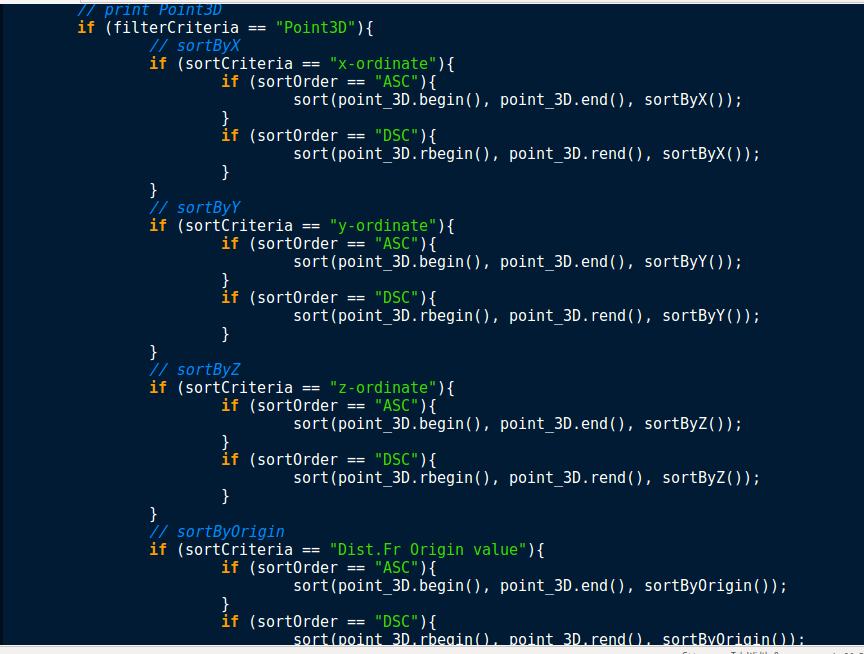
Option 5



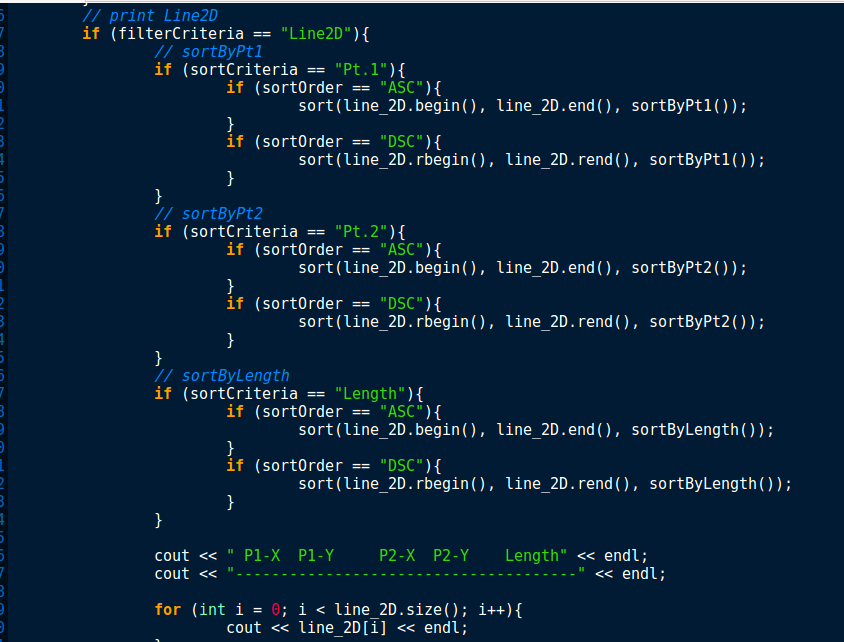
This is the function that will use for option 5. When user choose option 5, it will print out the filtering criteria, sorting criteria and sorting order that user choose in option 2, 3 and 4. If user didn’t choose in one of the options, it will print out the default criteria or order.



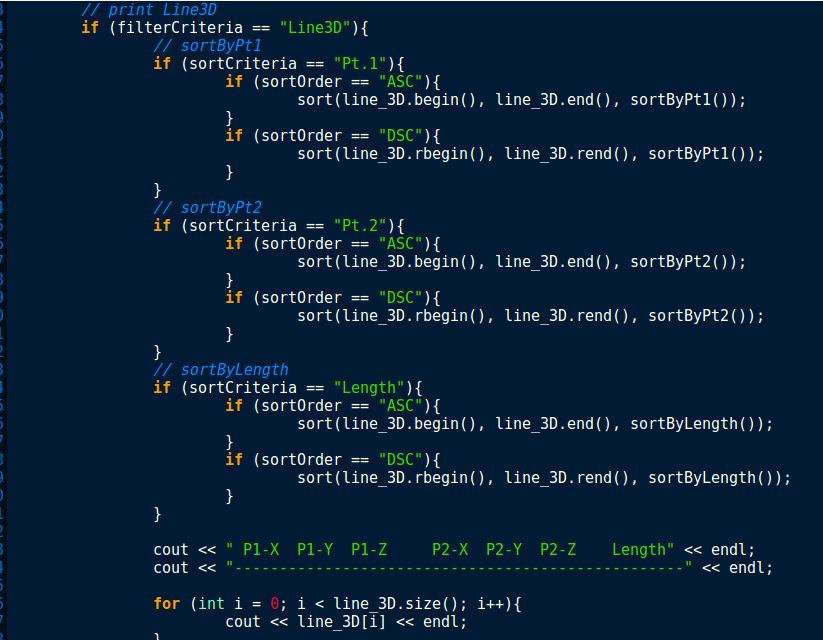
These are the conditions that if user choose Point2D for filtering criteria or didn’t select option 2, according to the options that user choose in option 3 and 4, the conditions will be different and each of the sorting criteria there will be a function to sort it. For example, if sorting criteria that user choose is y-ordinate, the program will sort the data by using function sortByY. Finally, the program will print out the data according to user needs.



These are the conditions that if user choose Point3D, it is same as Point2D but just add one more condition which is when user choose z-ordinate as their sorting criteria.

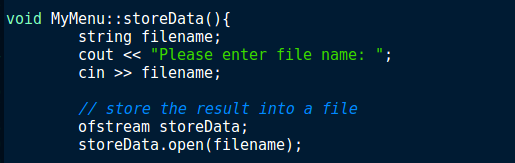


These are the conditions that if user choose Line2D for filtering criteria, according to the options that user choose in option 3 and 4, the conditions will be different and each of the sorting criteria there will be a function to sort it. For example, if sorting criteria that user choose is length, the program will sort the data by using function sortByLength. Finally, the program will print out the data according to user needs.

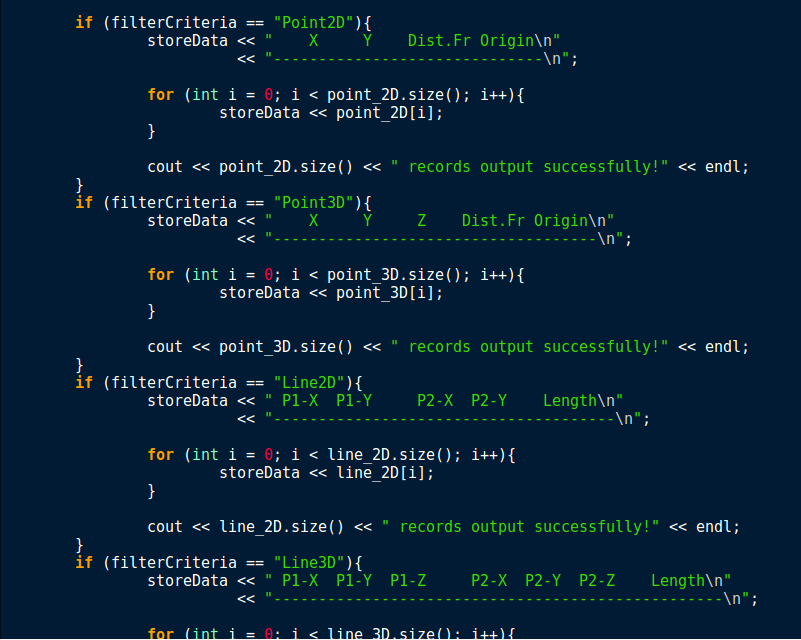


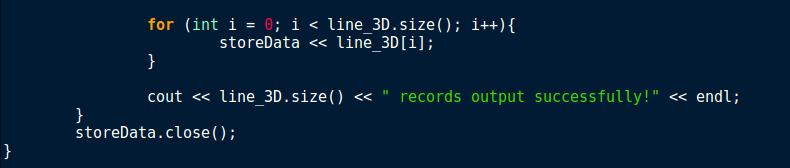
These are the conditions that if user choose Line3D, it is same as Line2D except that when print out the data, there will have two more column which is P1-Z and P2-Z.

Option 6



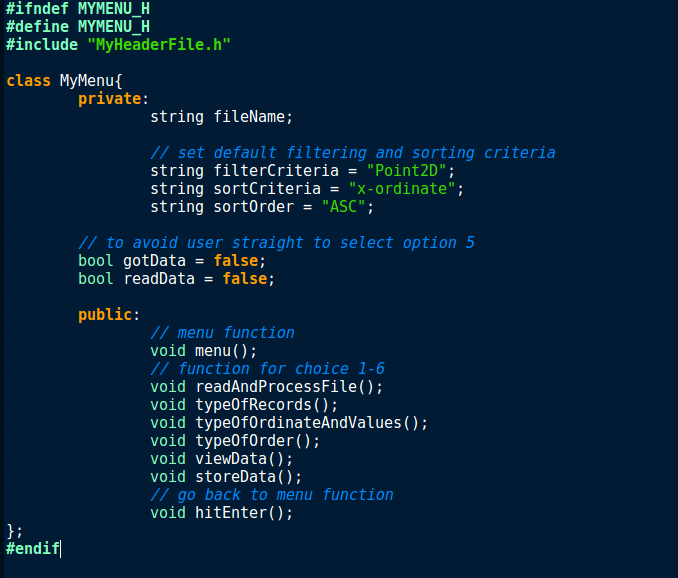
This is the function that will use for option 6. When user choose option 6, it will print out a message to ask user to enter a file name. This file name should be in new and not used for any txt file. After that, the program will store the data that into that file. The data that will be stored is the final data that the program show in option 5.





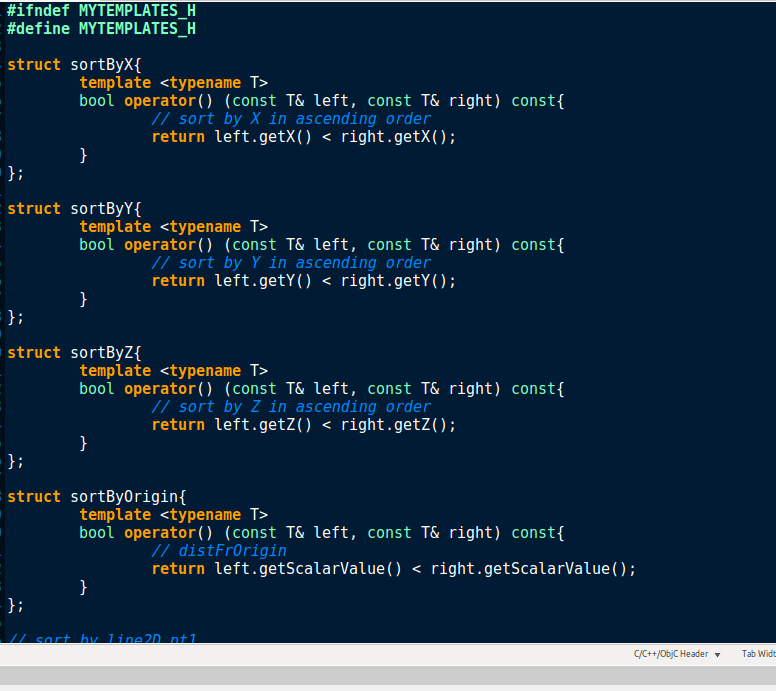
These are the conditions that will happen according to the filtering criteria that user choose. For example, if user choose Point2D for filtering criteria, the program will store all the Point2D’s data that after sorted into the file that user type in at beginning of option 6. After storing the data, the program will print out the success message, close the file and go back to menu.

MyMenu.h

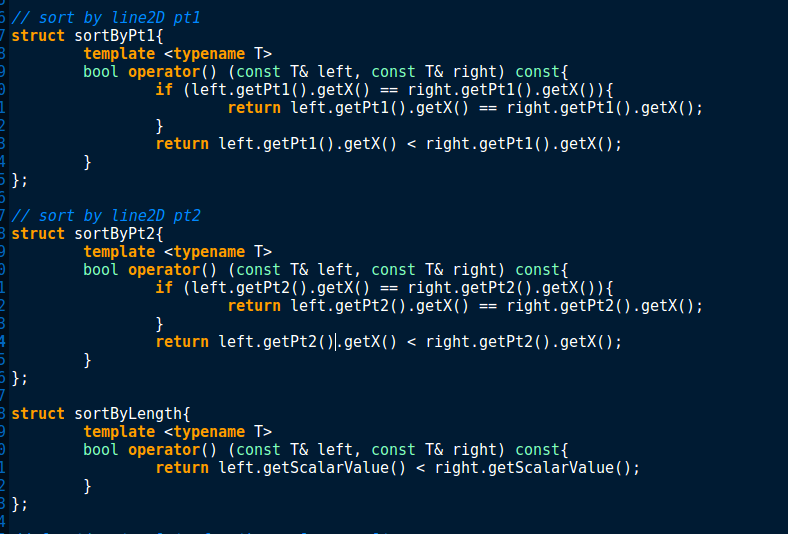


These are the include statement and header guards that I use in this file. Inside the class MyMenu, I have separate into two part which is private and public part. For private part, I have given string datatype to a variable call fileName and set the default for filtering criteria, sorting criteria and sorting order. To avoid user directly choose option 5 after user finish load data file in option 1, I have set two things to false which is gotData and readData. For the public part, these are the function that I use for menu, option 1 to 6 and hit any button to go back to menu.

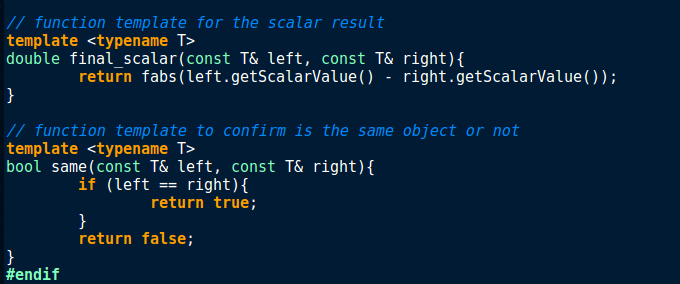
MyTemplates.h



These are the header guards that I use in this file. In this file, I list out some generic programming templates that I have used. I have using struct to group several related variables into one place. For sortByX, it used to sort by coordinate x in ascending order for all Point2D and Point3D data. For sortByY, it used to sort by coordinate y in ascending order for all Point2D and Point3D data. For sortByZ, it used to sort by coordinate z in ascending order for all Point3D data. For sortByOrigin, it used to sort by distance from origin for all Point2D and Point3D data.



For sortByPt1, it used to sort by point 1 for all Line2D and Line3D data. If coordinate X is same, sort by coordinate Y else sort by coordinate X. For sortByPt2, it used to sort by point 2 for all Line2D and Line3D data. If coordinate X is same, sort by coordinate Y else sort by coordinate X. For sortByLength, it used to sort by length for all Line2D and Line3D data.



For final\_scalar and same, they are a function template for calculate the difference of scalar value and to test for belonging to the same object.

Point2D.cpp

Text

Description automatically generated

These are the include statement that use in this file. For the default constructor, I have implemented two variables which are x and y and set them equal to 0. Next, I add two parameters which are x with integer datatype and y with integer datatype. Not only that, I also have included this x is equal to x and this y is equal to y and calculation for distance from origin. I have added a destructor in this file, destructor is a member function that is automatically called when an object goes out of scope or is deleted calling an explicit destroy. For the access function, I have implemented getX, getY and getScalarValue. Three of them will return x, y and distFrOrigin.

Text

Description automatically generated

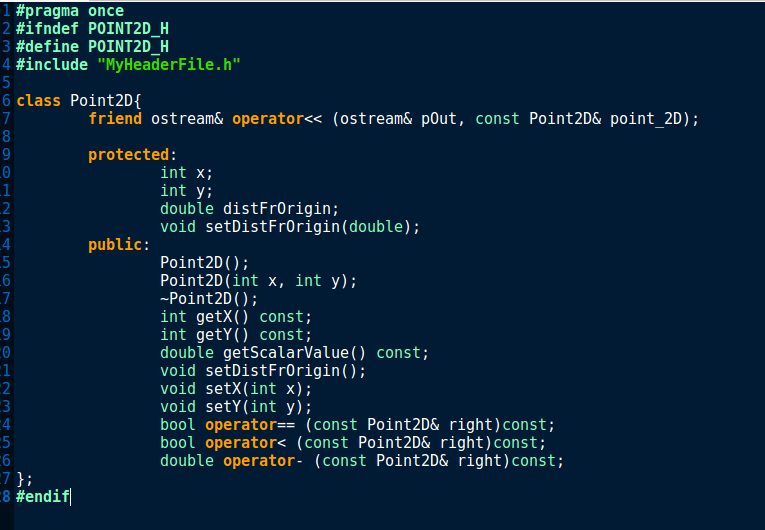
These are mutator function for this file. For setDistFrOrigin, I have given a formula for Point2D to calculate the distance from origin which is distFrOrigin. For setX, I have set this x equal to x and it also same as setY which is set this y equal to y.

Text

Description automatically generated

These are the overload function for this file, for the first boolean function, it uses to check equivalent function of Point2D data. For the second boolean function, it uses to sort the data of Point2D. For the double overload function, it uses to overload the difference of scalar value. For the last part, it uses to overload cout and the reason why I use pOut.flush instead of ostream.close is because ostream.close will cause the data loss so I use flush to prevent data loss happen.

Point2D.h



These are the preprocessor directive, include statement and header guards that I use in this file. Inside class Point2D, I have separate into three parts. First part is a friend function for Point2D, it is a function that can access private, protected, and public members of a class. For the protected part, I have implemented two integer datatype variable which are x and y, a double datatype variable which is distFrOrigin and a set function. For the public part, these are the constructor, destructor, access function, mutator function and overload function that implemented in Point2D.cpp.

Point3D.cpp

Text

Description automatically generated

These are the include statement that use in this file. For the default constructor, I have implemented a variable which are z and set it equal to 0. Next, I add three parameters which are x with integer datatype, y with integer datatype and z with integer datatype. Not only that, I also have included this z is equal to z and calculation for distance from origin. For x and y, I have implemented in Point2D.cpp so no need to rewrite again. I use setX(x) and setY(y) to get the value from Point2D.cpp. Next, I have added a destructor in this file, destructor is a member function that is automatically called when an object goes out of scope or is deleted calling an explicit destroy. For the access function, I have implemented getZ and getScalarValue. Two of them will return z and distFrOrigin.

Text

Description automatically generated

These are mutator function for this file. For setDistFrOrigin, I have given a formula for Point3D to calculate the distance from origin which is distFrOrigin. For setZ, I have set this z equal to z.

Text

Description automatically generated

These are the overload function for this file, for the first boolean function, it uses to check equivalent function of Point3D data. For the second boolean function, it uses to sort the data of Point3D. For the double overload function, it uses to overload the difference of scalar value. For the last part, it uses to overload cout and the reason why I use pOut.flush instead of ostream.close is because ostream.close will cause the data loss so I use flush to prevent data loss happen.

Point3D.h

Text

Description automatically generated

These are the preprocessor directive, include statement and header guards that I use in this file. Inside class Point3D, I have separate into three parts. First part is a friend function for Point3D, it is a function that can access private, protected, and public members of a class. For the protected part, I have implemented one integer datatype variable which is z and a set function. For the public part, these are the constructor, destructor, access function, mutator function and overload function that implemented in Point3D.cpp.

Line2D.cpp

Text

Description automatically generated

These are the include statement that use in this file. For the constructor, I add two parameters which are pt1 for Point2D and pt2 for Point2D. Not only that, I also have included this pt1 is equal to pt1 and this pt2 is equal to pt2 and calculation for length. Next, I have added a destructor in this file. Destructor is a member function that is automatically called when an object goes out of scope or is deleted calling an explicit destroy. For the access function, I have implemented getPt1, getPt2 and getScalarValue. Three of them will return pt1, pt2 and length.

Text

Description automatically generated

These are mutator function for this file. For setLength, I have given a formula for Line2D to calculate the length. For setPt1, I have set this pt1 equal to pt1 and it also same as setPt2 which is set this pt2 equal to pt2. Both are for point2D.

Text

Description automatically generated

These are the overload function for this file, for the first boolean function, it uses to check equivalent function of Line2D data. For the second boolean function, it uses to sort the data of Line2D. For the double overload function, it uses to overload the difference of scalar value. For the last part, it uses to overload cout and the reason why I use pOut.flush instead of ostream.close is because ostream.close will cause the data loss so I use flush to prevent data loss happen.

Line2D.h

Text

Description automatically generated

These are the include statement and header guards that I use in this file. Inside class Line2D, I have separate into four parts. First part is a friend function for Line2D, it is a function that can access private, protected, and public members of a class. For the protected part, I have implemented a double datatype variable which is length and a set function. For the private part, I have implemented pt1 and pt2 for Point2D. For the public part, these are the constructor, destructor, access function, mutator function and overload function that implemented in Line2D.cpp.

Line3D.cpp

Text

Description automatically generated

These are the include statement that use in this file. For the default constructor, I leave it as blank. For the constructor, I add some parameters which are pt1 for Point3D and pt2 for Point3D, pt1 and pt2 for Line2D, pt1 and pt2. Not only that, I also have included this pt1 is equal to pt1 and this pt2 is equal to pt2 and calculation for length. Next, I have added a destructor in this file. Destructor is a member function that is automatically called when an object goes out of scope or is deleted calling an explicit destroy. For the access function, I have implemented getPt1, getPt2 and getScalarValue. Three of them will return pt1, pt2 and length.

Text

Description automatically generated

These are mutator function for this file. For setLength, I have given a formula for Line3D to calculate the length. For setPt1, I have set this pt1 equal to pt1 and it also same as setPt2 which is set this pt2 equal to pt2. Both are for Point3D.

Text

Description automatically generated with medium confidence

These are the overload function for this file, for the first boolean function, it uses to check equivalent function of Line3D data. For the second boolean function, it uses to sort the data of Line3D. For the double overload function, it uses to overload the difference of scalar value. For the last part, it uses to overload cout and the reason why I use pOut.flush instead of ostream.close is because ostream.close will cause the data loss so I use flush to prevent data loss happen.

Line3D.h

Text

Description automatically generated

These are the include statement and header guards that I use in this file. Inside class Line3D, I have separate into four parts. First part is a friend function for Line3D, it is a function that can access private, protected, and public members of a class. For the protected part, I have implemented a double datatype variable which is length and a set function. For the private part, I have implemented pt1 and pt2 for Point3D. For the public part, these are the constructor, destructor, access function, mutator function and overload function that implemented in Line3D.cpp.