SENG2200 - Assignment 1

Cody Lewis - c3283349

March 24, 2018

Point

20 minutes of design, 10 minutes of implementation, 10 minutes of error correcting. The Point class was quite simple, as it is composed mostly of one line queries and mutators. I did however approach a design error with the to42F method (method converts doubles to 4.2f standard), the error was in the spacing, but was pretty quick located and corrected.

Node

10 minutes of design, 10 minutes of implementation, no error correcting The Node was the easiest implementation as it was almost a direct copy of what I implemented last year in the Data Stuctures course.

Queue

10 minutes of design, 20 minutes of implementation, 20 minutes of error correcting While a similar case to the node in that I remembered content covered in Data Stuctures, I utilized a simpler implementation of the Queue. This caused me to encounter some unexpected errors primarily being unable to return a reference from pop() which lead to the comprimise of splitting the function into pop() to remove and front() to get the front object.

ComparePoly

5 minutes of design, 10 minutes of implementation, no error correction The ComparePoly interface was very simple to design and implement as it is an abstract class, thus there was only the writing pure virtual functions and no true implementation.

Polygon

20 minutes of design, 2 hours of implementation, 1 hour of error correcting The Polygon object was not very hard to implement, but I experienced major errors when implementing the MyPolgons class. I chose to use a Queue to store the Points in this class as the points are always accessed sequentially, store in order and are dynamically added. While the Polygon was mostly a simple wrapper for the Queue, utilizing some functions without breaking encapsulation or creating memory leaks took a bit of thought. Examples of such would be the toString() and findArea() classes both of which use a copy of the points Queue in order to avoid this problem.

MyPolygons

1 hour of design, 4 hours of implementation, 6 hours of error correcting This class caused me the most trouble, I anticipated that the circular doubly linked list to be most difficult part thus I spent a decent amount of time designing it. The amount time spent designing it made the implementation of the list mostly trivial, although the were initially a few design error with the destructor and the insert function. For the main input I used the overloaded operator»() and string manipulation methods which made it decently easy with only a few design errors occurring to me with the implementation. The main troubles encounter with this class were a number of implementation errors upon using this class to hold Polygons, the primary error being with how the Node operator»() works. In the Node, the method

operator»() creates a new instance of the contents then uses that contents operator»(), while this method usually works well, this also means the contents destructor is called at the end of the method in order to free the blocks used in the instance of the object in the method. As the object sent into the contents of the node is reference to instance used in the method destruction of the instance also means destuction of the Queue of points that is pointed to by the Polygons objects, causing the Polygon to have a pointer pointing to an irrelevant memory address. This caused me to spend a lot of time locating and attempting to correct the error, I eventually found the solution in adding the modData() method to Node which allows for adding the data to the Polygon without it destructing. The next major point of error I had was the copy constructor for MyPolygons which creates a copy list which is sorted. The initial difficulty was avoiding the destructor error while still doing a deep copy of the list (so the original list is not modified) this still proved be decently easy to do once I decide to utilize similar problem solving to the input case. What was more difficult the sorting, using the overloaded operator greater than in Polygon made the code simpler but for a while I attempted to sort the list after its copy which created a mess of pointers. I managed to figure out that an easier form of implement would be doing an insertion sort as the Nodes are being inserted.

PolyDemo

20 minutes of design, 15 minutes of implementation,4 hours 10 minutes of error correcting The demo method was one of the easiest classes to implement with the main error I approached being with the interface, I found that in the switch statement that I had seperate the copy pointer MyPolygon construction had to be separated out into a individual method. Three days before the due data of the Assignment I found that a segmentation fault occurs at ES409, the segmentation fault occurs due to the adding methods which take a Polygon reference as the argument. When the data from such Nodes is called, the program segmentation faults at ES409. After some error correcting using valgrind I found that the error arised from a bad conditional statment, after correction the program no longer produces segmentation faults when compiled with g++ 4.9. The updated assignment notes meant I had to go back and edit the to42F and to52F methods as I misinterpreted the specifications.

The errors approached in the project were 80% implementation and 20% design.

For the cases of rectangles and squares, inheritance may be used. The rectangle and square could inherit the Polygon class and have a modified findArea() method, which multiplies the length of two sides. All other methods may stay the same.