Evaluation

My map is a fully functional implementation of a map, using operator overloading to make it more intuitive to use. However, some of its functions are inefficient in the way they handle data, such as the findByKey method, which uses a linear search rather than the binary search used by the stl map class. As such, my map does not take full advantage of the map data structure and it can perform very poorly in comparison in intensive programs.

However, the erase method of my map is more intuitive than that of the stl map, as mine only requires a key, whereas the stl map requires a pointer to the element you wish to erase. The printmap function can be quite useful, and is a feature not offered by the stl map, however it is not type safe, and as such will cause an exception if the datatypes of the map do not have an overloaded << operator to output information.

However, by not providing access to the key data in the map, I prevent users from accidently altering it and causing problems with it. The key values can still, unfortunately, be accessed through the use of iterators, and as such user’s code could use them to alter the key values in such a way as to compromise the map structure, as there are no checks made to ensure that a change to a key value is permissible (i.e., the new value is not already in the map).

The main way to improve my class would be to improve the search function, as a linear search is an order of magnitude slower than a binary search on large amounts of data. Also, the print function would need to be type safe in some way, such as by only allowing it to be accessed if the datatypes of the map had an overloaded << function.

In general, c++ does an excellent job for abstract data types, as evidenced by the power of the standard template library. It mainly does this through templates, which allow generic algorithms and datatypes to process any appropriate data structure passed to it, and allows for much more robust and reusable code.