Object Oriented - Map Class Report

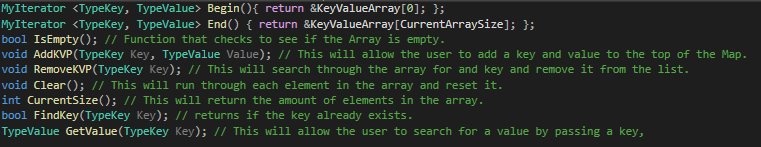
# Introduction

In this report I will be discussing the map class I have implemented in C++, I will talk about the design decision I have made while making this map class as well as the Weakness of my map class and improvement I could have made.

I will also talk about the support that C++ offers when it comes to working with ADT’s, I will also compare my Map class to the Map class in STL.

# Design Decisions

Interface Methods: For my Interface methods I went for the following methods:



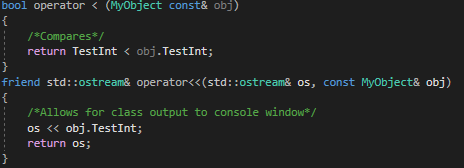
I chose these methods as these were the most useful methods that I thought a base map class should have.

Private Methods: for my private methods I went for the following methods:



This was because these two methods are critical to the map as it needs dynamic memory and it needs to be sorted.

Friends and overloads:

When a user wants to use a user defined class they have to provide a < operator and << friend operator, this was because it was the best way I could get the implementation working at that time. They have to provide a < because this allows me to also use it as an equal and not have to make them provide another operator. They have to provide a << because this it will allow them to output to the screen, this is a friend function because it doesn’t directly have access to ostream so the friend keywork allows for this function to use it.

# Weakness and Improvements

**Weakness**

Forced Operator Overloading: One of the weaknesses of my map class is that if a user wants to use a user defined class then they have to provide two overload operators for the < operator and << operator.

Efficient: My Map class isn’t as efficient as I would like it to be, if I was going for a very efficient map class I would have used Arrays for data storage and would have gone for a better sort function.

**Improvement**

If I wanted to improve my map class I would have opted for different approaches for data storage and sorting as this is the part of my Map which are the least efficient part but I would also like to find a work around the operator overloading as this could cause problems for user that haven’t read the documentation.

If I had any alternative representations it would have been to make the class easier to use as in remove the operator overloads and to make it easier to access by adding functions to get data like the [] operator.

# Abstract Data Type Support

C++ is really useful when it comes to supporting ADT’s, Templating is C++’s way of allowing us as programmers to make functions and classes that can support any data type possible. A good example of this is my very own test function which is a template function. The template function takes two arrays of any data type and will run through all the test programs without breaking. It would have taken me multiple function to get it working without templating which is more function and more work. Template was also implemented in my map class so that any data type could be added into it.

# Map class vs STL Map

The STL Map class is way better then my implementation as this is due to the fact it has more functions as it allows the user to be flexible on how they want the map to show their data. My Map class allows for the use of Iterators and User defined classes as does the STL Map but the STL gives the user more control over how they could use the Iterator.

The STL map allows the user to access elements by using operators like [] and At while my map class doesn’t allow for this. So, my Map class and the STL Map are near enough the same and what it actually does but the STL does allow for the user to be more flexible with the Map.