Object Oriented - Map Class Report

## Design Decisions

All the function in my Map class have been templated, this is because it will allow for all data types to use the map class. I also used templating for my test function as this would reduce the amount of copy and pasting needed for the test cases.

Public Method:

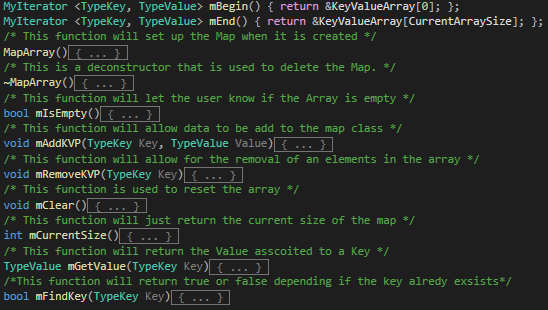
Begin – This function is used for iterators, this will simply return the start of the array, this was critical for iterators to work.

End – This function is used for iterators, this will simply return the last element of the array, this was critical for iterators to work.

IsEmpty – This function is a Boolean, this will check to see if the array size is empty or not, this was added because it allows the user to check the map is empty.

AddKVP – This function will allow the user to add a key and value to the map this is because it’s a critical function for the map class, If the user adds a key that already exists it will override the value at that key if not it will just add the key and value. Once a key and value has been added it will sort the map and if memory needs more allocation it will call the add memory function.

RemoveKVP – This function will allow the user to remove a key and value from the array by providing a key this is important because the user needs to remove keys and values from the map. In this function it isolates the key and value from the array, once isolated I pass all the new data into a new array and sort it.

Clear – This function will clear all the elements in the array, this is important because it makes to easier to clear the array. In this function I simply delete the array and create a new one.

CurrentSize – This function returns the current number of elements in the array, this is needed to see how many elements are in the array.

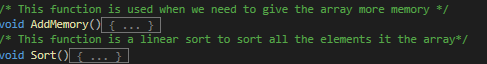
FindKey – This function will return true or false to see if the key is already in the map. This is important as it allows the user to search the map for a key. This simply just run through all the keys and returns true or false;

GetValue – This will return a value in the map, this will allow the user to get a value tied to a key, this will run through all the keys and if it matches it will return the value.

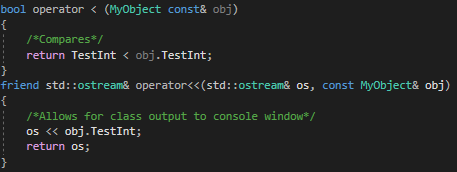
Private Method:

Sort – This will sort the map by keys, this is a private method because the map gets sorted once a key and value has been added and when a key and value is removed so there is no need for the user to have access to it. This sort is a simple linear sort as I’m not to bothered about efficient.

AddMemory – This will give the array more memory, this is a private method because the user shouldn’t have to increase the memory as the map already does it when it needs to. This will simply increase the map amount by times the current amount. I transfer all the new data to the new array and delete the old one and assign the pointer to it.



Operator Overloading:

For the user to get User defined classes working they have to provide me with two operator overloads for “<” and “<<” as one is used for comparison and the other is used to output to the screen with iterators.

This was the only way I could get it working in the map class and is a big weakness with my map class and I wish I could have done it a better way.

Another weakness of my map class is the efficient of it, since efficient wasn’t a big part of the implementation requirements I decided to just not do things in the most efficient way. If I had to improve the map class I would make the program more efficient and fix the operator overloading for “<” and “<<”.

For the << operator to work it will need to be a friend of the Ostream class, but the << operator is only ever need if the user plans to output to the console window so it might not be needed most of the time.

The friend declaration was used when gaining access to the Ostream class, this allows for the overloading of the << operator.

## Dynamic allocated Arrays

When it came to allowing the user to have an array of any size I had to resize the array, the only way to do this was to add all the data to a temp array and delete the old array and then make the old array pointer point at the new array. This wasn’t the best way to do this as you have to loop through every element of the array to just remove or add elements into the array.

## \_CrtDumpMemoryLeaks

When it came to testing my map class I used a function called \_CrtDumpMemoryLeaks which allow for me to see where memory leaks have been created so it allowed me to make a safe memory map class.

At the end of every test case I use this function to see if any memory leaks have been created while I have tested the function. This will output to the console window if a memory leak has been created or not.

## 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 class has a big focus on efficiency as it has a few security problems, a good example is the clear where it just resets the element count but doesn’t delete any data, my map class is less efficiency but focuses on having a secure map class so everything is safe.

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 when it comes to functionality.