Compulsory 1- Rapport

Anders P. Åsbø

For this assignment, I collaborated with my group partner, Thomas Dalstein Sæther, and the code we are handing in was made together. The source code for this project can be found in this GitHub repository.

I found it relatively easy to implement a dynamic array, since I have done it before (see this GitHub repository I worked on during the summer break). However, we decided to make our new implementation using memcpy and memmove instead of for-loops, when handling moving of data and reallocation, to give ourselves a challenge. While we implemented the dynamic array as a template class to allow different data types, we discovered that memcpy and memmove only supports primitive datatypes like integer types, char, bool and floating point numbers, because they do not handle the extra data needed by class-based datatypes like std::vector, and std::string.

I tried bypassing the memcpy/memmove problems by having the internal standard data array of our class, be an array of pointers to the actual elements. However, this resulted in a compiler error stating that memcpy and memmove did not have suitable overloads for handling pointer types, and I found no info on how to fix this. Instead, I tried using a wrapper struct-around each element, but this did not work either. So currently, our DynamicArray-class only supports elements of primitive datatype. To make it support more complex datatypes, I believe we need to use for-loops instead of memcpy and memmove.

For the sorting algorithms, we decided to have them as a separate library in the "Sorters.h" header, with the namespace Sorters, so to not bloat our DynamicArray-class. The sorting functions can be called directly with the array to be sorted passed as reference. I mostly worked on the dynamic array itself, while Thomas did most of the sorting algorithms. However, we helped each other understand and debug what the other was working on. In addition, Thomas did the search algorithms in the DynamicArray-class, while I implemented the heap-sort algorithm, and helped make Thomas' sorting functions template based.

To complete the tasks, I primarily used the links provided in the assignment text, but to learn how to build a min-heap from an array based binary tree, I used this web-page (Educative Answers Team, 2022), and modified it to build a max-heap instead for the heap sort algorithm.

References:

Educative Answers Team. (2022). *How to build a heap from an array*. Educative: Interactive Courses for Software Developers. https://www.educative.io/answers/how-to-build-a-heap-from-an-array