

Data Structures and Algorithms

**Lab 1: DynArray.h**

**The Scenario**

Your boss has tasked you with creating a Dynamic Array from scratch in C++ to be included in the latest library the company publishes. He believes in you and your ability with C++, Pointers, and Coding! If you succeed, your salary will be upped to $1,000,000,000 per year, and you’ll be given full access to the company yacht.

Dynamic Arrays are analogous to vectors in the Standard Template Library. They include functionality such as **random access**, **resizability**, and **adding/subtracting elements**. One key thing to remember is that they are, quite simply, Arrays with added functionality. Your job is to implement this added functionality, including custom operator functionality and new functions.

**What To Do…**

Open DSA\_Defines.h. Find the lab for DynArray. Where it says “#define” at the top of each block of text, turn the 0 into a 1 to enable the lab. You can turn each test individually on or off. In some labs it may benefit you to go in order, and in others it may benefit you to jump around. i.e. if you use your Assignment Operator in your Copy Constructor, you’ll want to make sure it passes the Assignment Operator test first.

Open DynArray.h. There will be instructions written in the comments on what is expected. In programming, there are an infinite number of ways to write the same function and this class may be your first exposure to that. The instructions will be slightly more vague than you may be used to. In a career, you aren’t given explicit instructions as often as a general description of what is wanted/required. Below is the gist of each function and variable.

***Variables:***

**mSize** How much data there is in the dynamic array. (i.e. how many slots in the internal array are used.)

**mCapacity** The actual capacity of the array (i.e. how many slots exist. Not just how many are used.)

***Functions:***

**Default Constructor** This will create your dynamic array to either a default capacity of 0 or to what the user passes in.

**Destructor** This deletes the object and clears up any dynamically allocated memory.

**Copy Constructor** Creates a copy of the object passed in.

**Assignment Operator** Overloads **=** to set two DynArrays equal.

**Access Operator** Overloads **[ ]** to give access to elements of the internal array.

**Size** Returns thenumber of elements in the array.

**Capacity** Returns the overall capacity of the array.

**Clear** Resets everything to if the default constructor had been called with no parameters. Cleans up any Dynamically Allocated Memory.

**Append** Stores an element in the next available space in the DynArray. Resizes the internal array to be larger if needbe.

**Reserve** Resizes the internal array based on the parameter. If it is less than the current capacity, do nothing. If 0 is passed in, double the array size. Special Case: If 0 is passed in, but mCapacity is 0, rather than doubling the 0, simply set mCapacity to 1. This function also copies over all the data to the new array.

**Tips, Tricks, and Resources**

Functions and data members available in the Array class ( i.e. myArray.size() ) can be found on the Cplusplus.com documentation for arrays: <http://www.cplusplus.com/reference/array/array/>

**Plagiarism**

Plagiarism and Academic Dishonesty are considered a **very** serious offense in this class and can have a range of consequences including suspension, and in very serious cases, expulsion. If you either share your code or copy someone else’s code, you will be given a **0** on your lab and can face further disciplinary action.

In other words, don’t cheat please!