Jacob Greer / Jonathan Wilson

CSIS 215-001

Feb. 19th, 2021

**Approach – Lab 1 – Bag ADT**

We decided to use a vector over an array due to the nice amount of built-in functions it includes. For example, all it takes to empty a bag is by calling [vector].clear(), which empties the contents of the bag. We were aware of the implications of doing this (having to deal with smart pointers) but this made the first part of the project go a lot smoother.

For our Boolean functions, we learned from 112 that it is best practice to not directly return “true” or “false” but instead pass the value to a separate Boolean and returning that instead.

In our for loops, using ++i over i++, according to Dr. Poole, is slightly faster/more efficient, so we do that wherever possible.

* ABag.h
  + ABag(), ~ABag() are the constructor and destructor. maxItems dictates the max capacity of the bag (for the sake of this assignment). ~ABag() deallocates memory with the vector swap function.
  + addItem
    - If there is room in the bag (compared to maxItems), insert a new entry into the bag. Our approach compares the vector’s size to number of max items and uses the push\_back function of vector to add an item to the bag.
  + Remove
    - Our remove function successfully removes items and also takes into account the fact that we start counting at zero, with the second “else if” statement.
  + removeTop
    - removeTop’s hasRoom boolean only works if the vector is larger than 0. If it’s larger than zero, we use the erase function of vectors to erase the end part of the bag (data.end() – 1).
  + Find
    - The find function simply rolls through the vector with a for loop to compare the returnValue with data.at(i), returning true.
  + inspectTop
    - inspectTop makes a simple comparison, first determining if the bag size is 0 (no top) and then changing the passed-in item’s value to the value of the top item.
  + emptyBag()
    - One of the reasons we used a vector was for functions like data.clear(), which we used here to empty the bag. This seems more efficient than a for loop.
  + += operator
    - The operator points back to the addItem function, allowing us to use the overloaded += operator in main.
  + Size()
    - Another reason we used a vector was to take advantage of data.size(), which returns the size of the vector, though this wouldn’t be hard with an array either.
  + bagCapacity()
    - This returns maxItems + 1, a value we dictate at the top of ABag.h.
    - We use this function in the insert() function of BDictionary to determine if there is space in the bag.
* BDictionary.h
  + BDictionary, ~BDictionary are the constructor and destructor. ~BDictionary deallocates dictionary from memory with the delete function.
  + Clear()
    - Calls the emptyBag function from ABag, deleting all objects in the bag vector.
  + Insert()
    - Inserts a new KVpair (key and value) pair into the bag. **The unique\_ptr is the smart pointer in our program**, utilizing the <memory> library.
    - We call the bagCapacity() function from ABag to determine if there is enough space in the bag.
  + Remove()
    - Removes dictionary records. **The second smart pointer (unique\_ptr) is here.** We remove an object in the bag whose keyvalue pair matches the smart pointer that was passed in.
  + removeAny()
    - **The third smart pointer (unique\_ptr) is here**. We set the kvpair to the top value of the bag which then gets removed from the dictionary. If this returns true, the bag was not empty and the kvpair is properly updated.
  + Find()
    - **The fourth smart pointer (unique\_ptr) is here.** We call the ABag find function to search for an object of the same key value pair.
  + Size()
    - Simply returns the size of the dictionary, calling the size function from our ABag implementation.
* Bagtestmain additional tests
  + We tested some ABag.h functions that were not demonstrated in the default bagtestmain. We demonstrate that inspectTop works by calling it in the same line as our cout statement. If it works, it outputs a value of 1. The same goes for our size() function – it outputs the value before and after a change was made to it, demonstrating that the size has increased.