

Jerry Lee  
CMPE126  
21 February 2019

### Lab 3: Arrays

#### Running the code

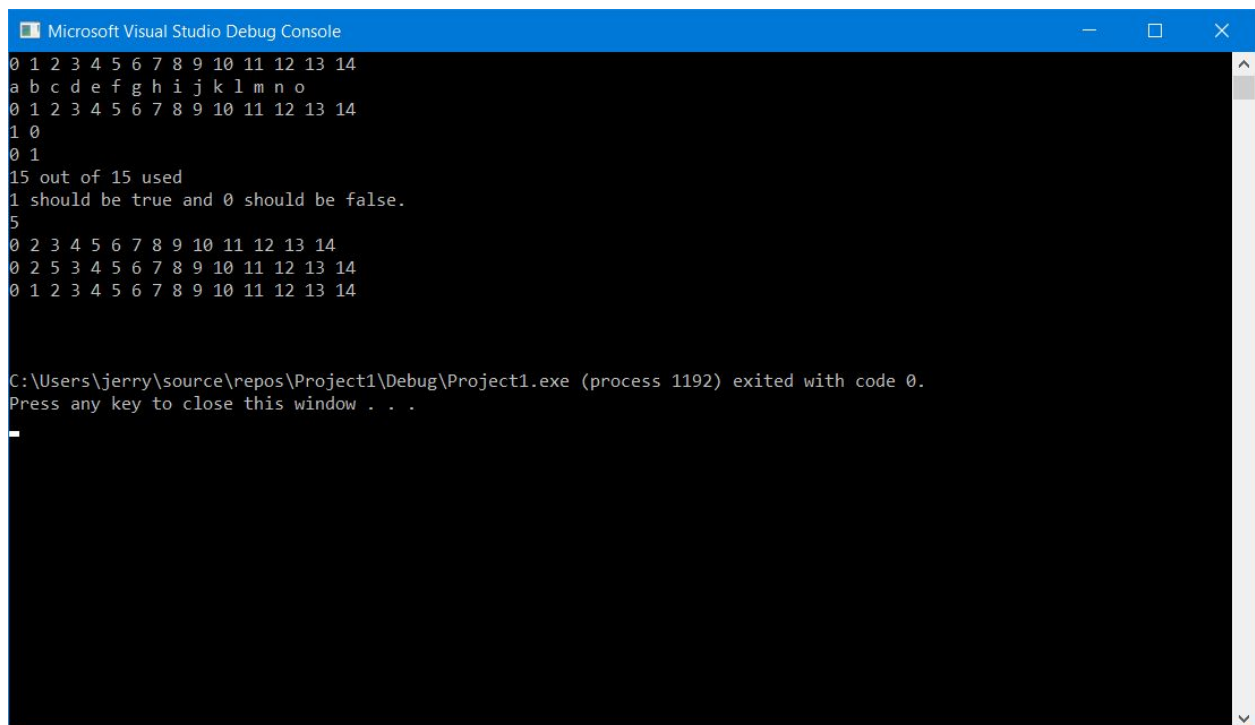
The first three lines are for populating 3 arrays - first 2 with insertEnd and last one with insertAt. The next two lines test the functions isFull and isEmpty with a full arraylist and an empty arraylist.

The next line checks the arraylist size and the max arraylist size.

The next line checks isItemEqual with a true case and a false case.

The next line retrieves the element at position 5.

The next three lines uses removeat, then insertat, then replaceat. Then, the list is cleared.



```
Microsoft Visual Studio Debug Console
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
a b c d e f g h i j k l m n o
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
1 0
0 1
15 out of 15 used
1 should be true and 0 should be false.
5
0 2 3 4 5 6 7 8 9 10 11 12 13 14
0 2 5 3 4 5 6 7 8 9 10 11 12 13 14
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

C:\Users\jerry\source\repos\Project1\Debug\Project1.exe (process 1192) exited with code 0.
Press any key to close this window . . .
```

#### Time complexity

Time complexity is a measure of how long, or how efficient, our code works. In this lab, we defined an ADT called arrayList and implemented numerous features. The time complexity of each feature is listed below.

#### Functions

- 1) isEmpty:  $O(1)$ 
  - a) This function determines if an arraylist is empty. It checks if size is equal to zero or not. Therefore, it is constant time
- 2) isFull:  $O(1)$

- a) This function determines if an arraylist is full. It checks if the size is equal to the limit so it is constant time.
- 3) `listSize`:  $O(1)$ 
  - a) This function fetches the variable size. It checks how big the arraylist is. Thus, it is constant time.
- 4) `maxListSize`:  $O(1)$ 
  - a) This function fetches the variable limit. It checks how big the arraylist can become. Thus, it is constant time.
- 5) `Print`:  $O(n)$ 
  - a) This function prints every element of the arraylist. In order to do so, I used a for loop to fetch each element. The size of the for loop depends on how big the arraylist is. Thus, this function is linear time.
- 6) `isItemAtEqual`:  $O(1)$ 
  - a) This function checks if the element at a position is equal to a given input. It only requires one comparison so this function is constant.
- 7) `insertAt`:  $O(n)$ 
  - a) `insertAt` inserts an element in a given position in the arraylist. In order to do so, I used a for loop starting from the end of the arraylist to the desired position. The loop shifted every element right by 1 to make space for the spot where the new element would be placed. The amount of times the for loop executes is dependent on how big the arraylist is and what position the element should be placed. Thus, it is on linear time.
- 8) `insertEnd`:  $O(1)$ 
  - a) This function inserts an element at the last position in the arraylist. Thus, it is constant.
- 9) `removeAt`:  $O(n)$ 
  - a) This function removes the element at position  $n$ . In order to do so, The element at position  $n$  is replaced with the element at position at  $n+1$  and so on until the loop reaches the end of the arraylist. Thus, this function is on linear time.
- 10) `retrieveAt`:  $O(1)$ 
  - a) This function returns the element at a given position. Thus, it is on constant time.
- 11) `replaceAt`:  $O(1)$ 
  - a) This function replaces an element in the arraylist. Thus, it is on constant time.
- 12) `clearList`:  $O(1)$ 
  - a) This function sets the size of the arraylist to 0. Thus, it is on constant time.
- 13) `operator=`:  $O(n)$ 
  - a) This function deletes the old arraylist and builds a completely new one. A for loop is required to build the new arraylist. Thus, this operator is on constant time.
- 14) `DoubleSize`:  $O(n)$ 
  - a) This function doubles the size of the arraylist in case there is not enough space. In order to do so, A temporary arraylist is created with double the size of the old arraylist. The elements of the old arraylist are then transferred to the new one. Then, the old arraylist is deleted and the pointer is rerouted to the new arraylist.

Since a for loop is required to copy the elements of the first arraylist to the second arraylist, this function is on linear time.