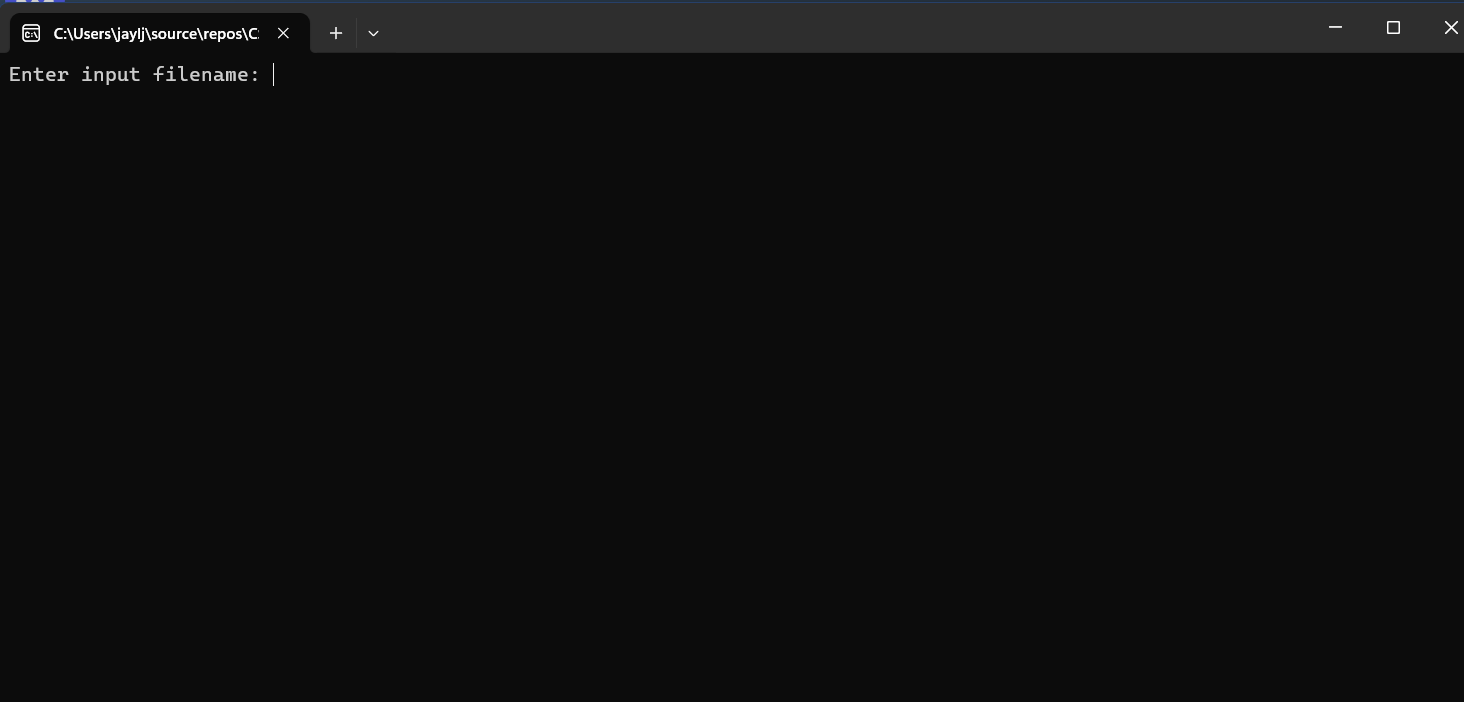
Hello, I am Jaylen Jones, and this is my first assignment for Comp-Sci 303 Data Structures. This program's main purpose is to be able to read data from an input file into an array with functions that allow you, the user, to

* Modify the index (adding, removing, modifying values…)
* Find integers within the index
* Display the array  
  But we’ll go a bit in-depth.



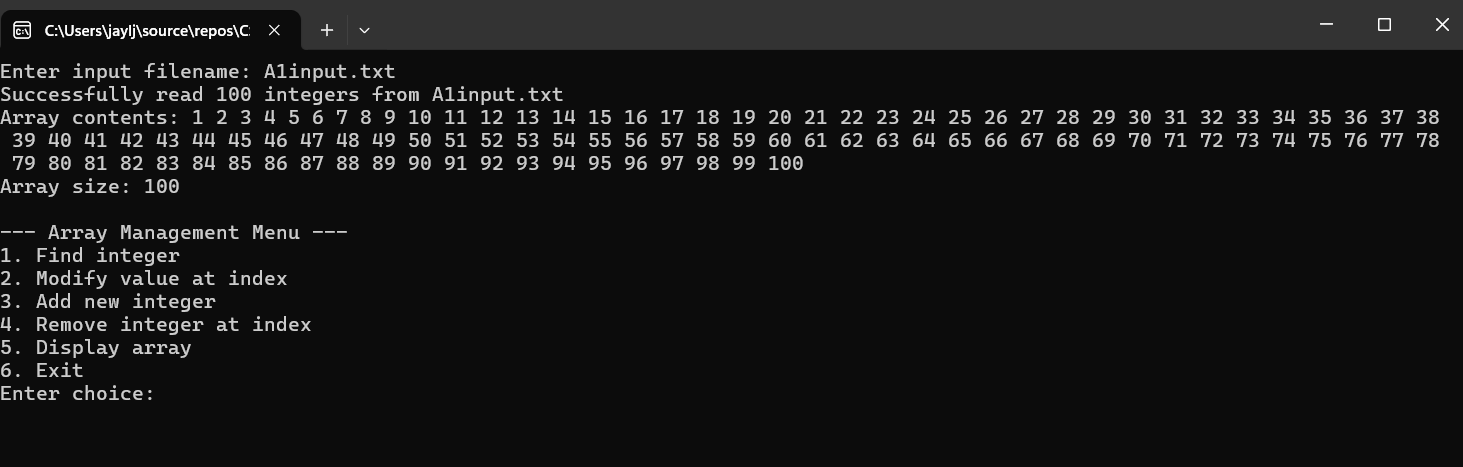
When you start the program, it’ll first ask you to enter the input file name.

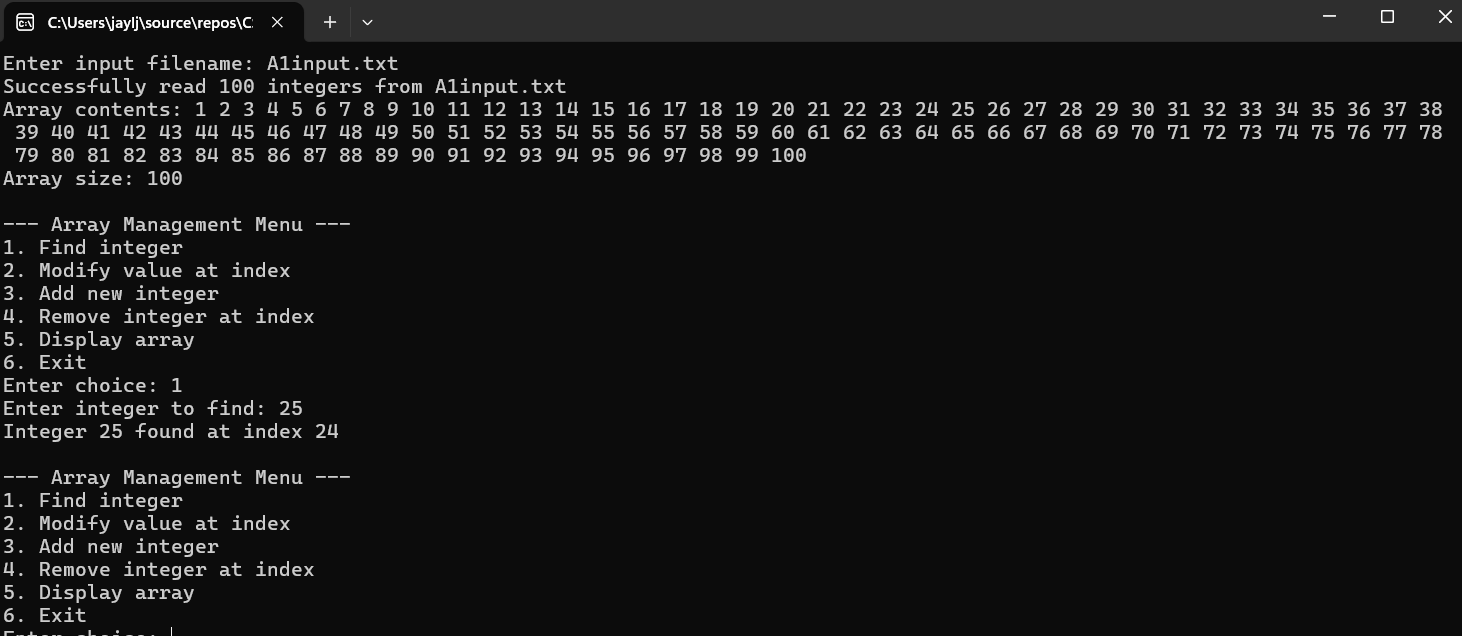
And when you put in the right file name, the program will read the file to check if everything is right, the program will tell you that it went successfully and show the array contents and its size.  
The numbers index/place always: start off at 0. So, the index of 1 is 0, 2 is 1, etc… like this:

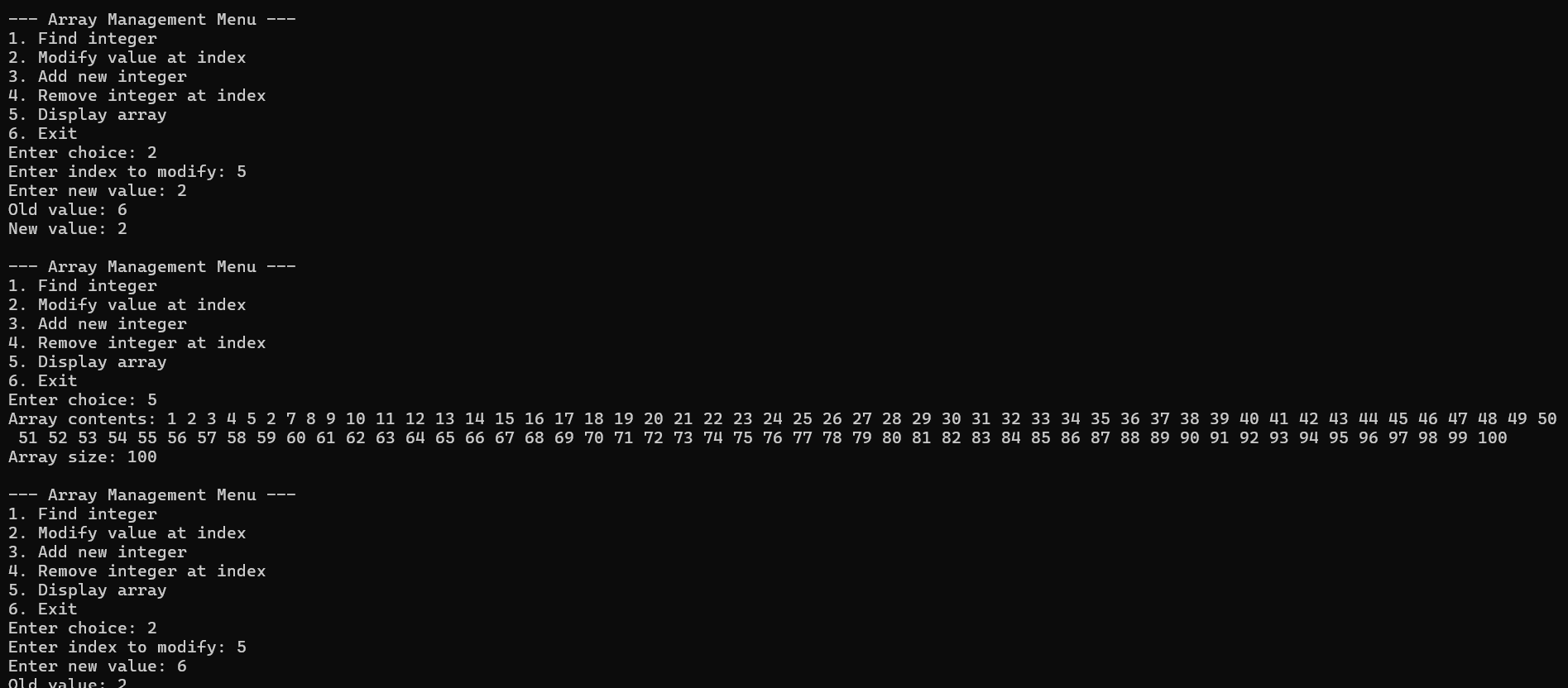
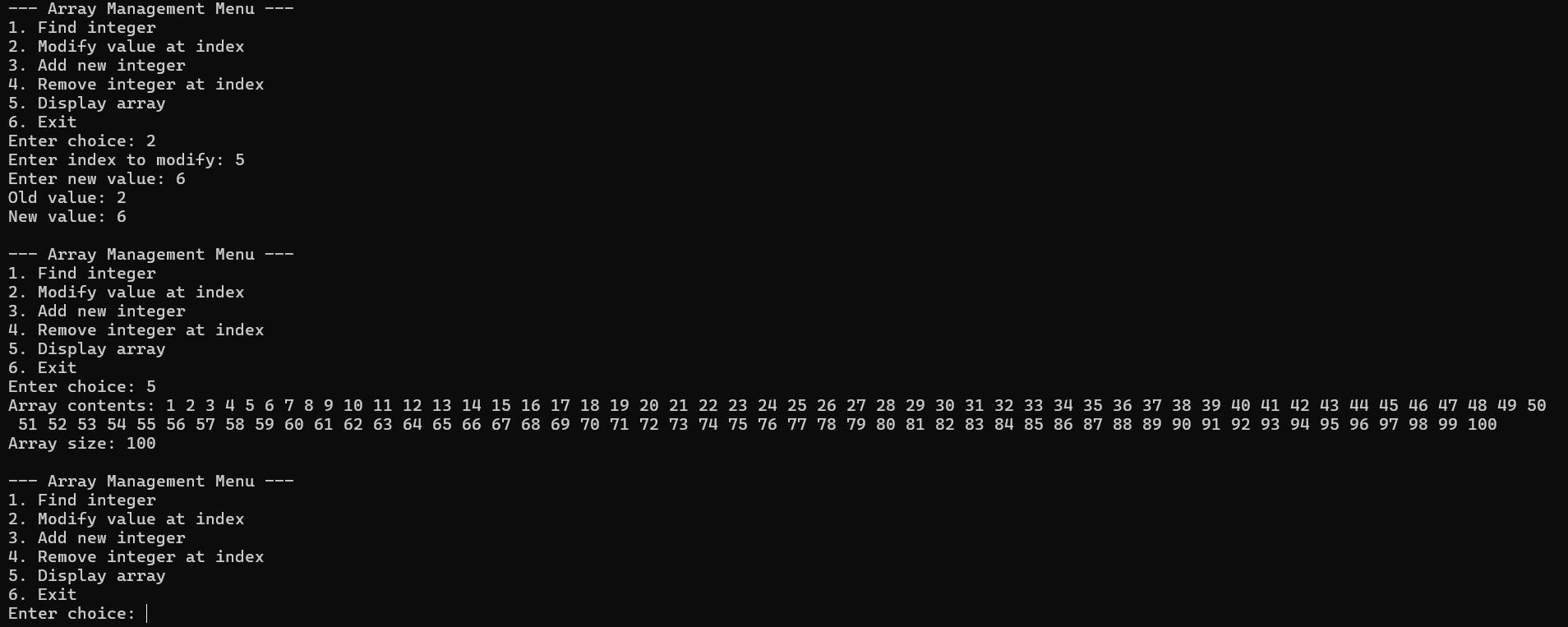
| Index | 0 | 1 | 2 | 3 |
| --- | --- | --- | --- | --- |
| Values/Numbers | 1 | 2 | 3 | 4 |

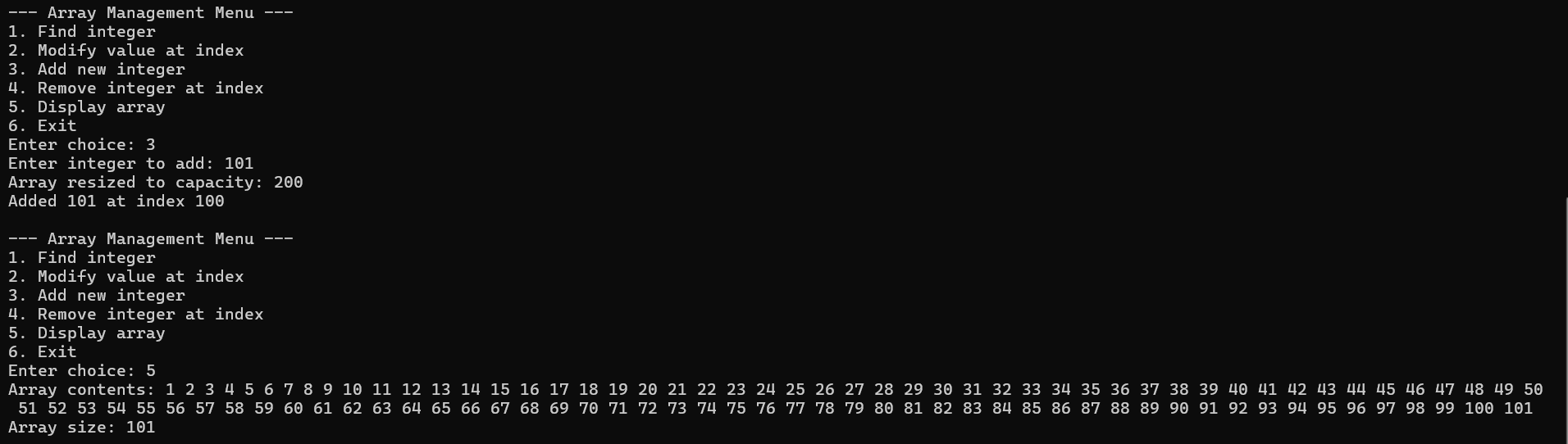
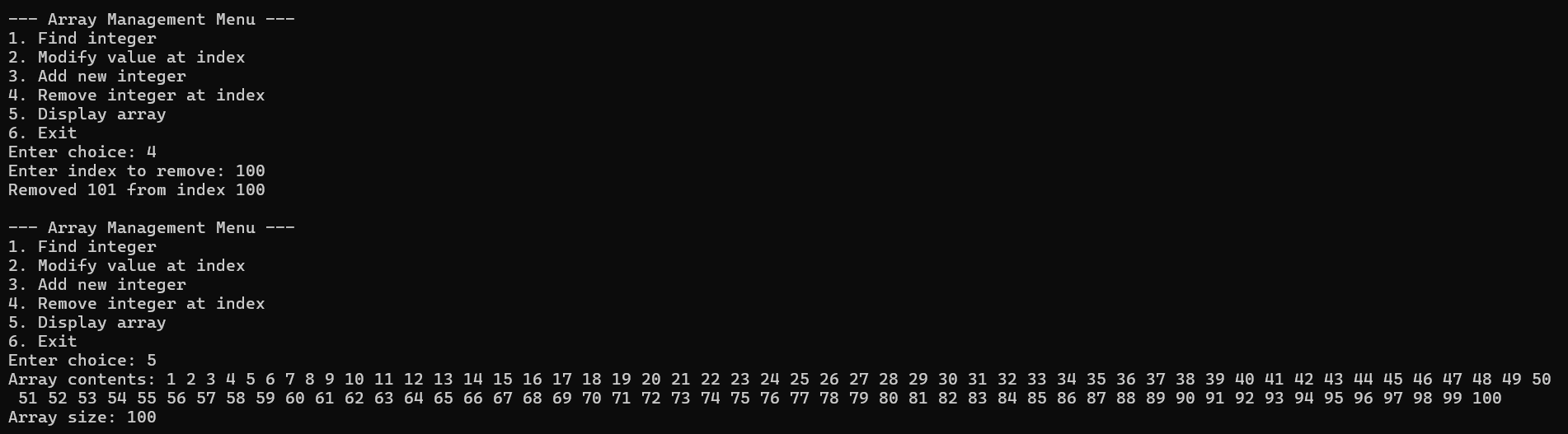
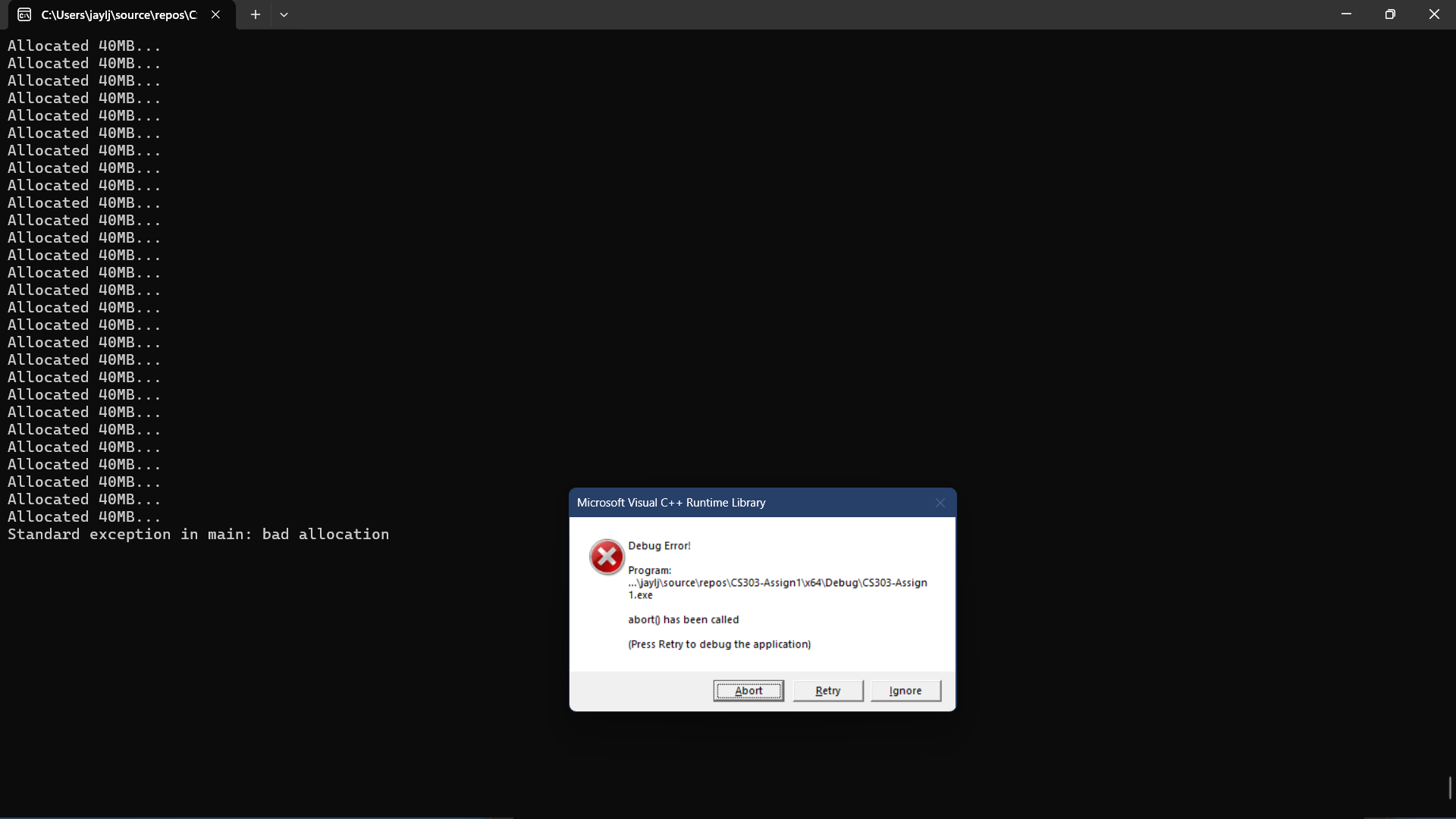
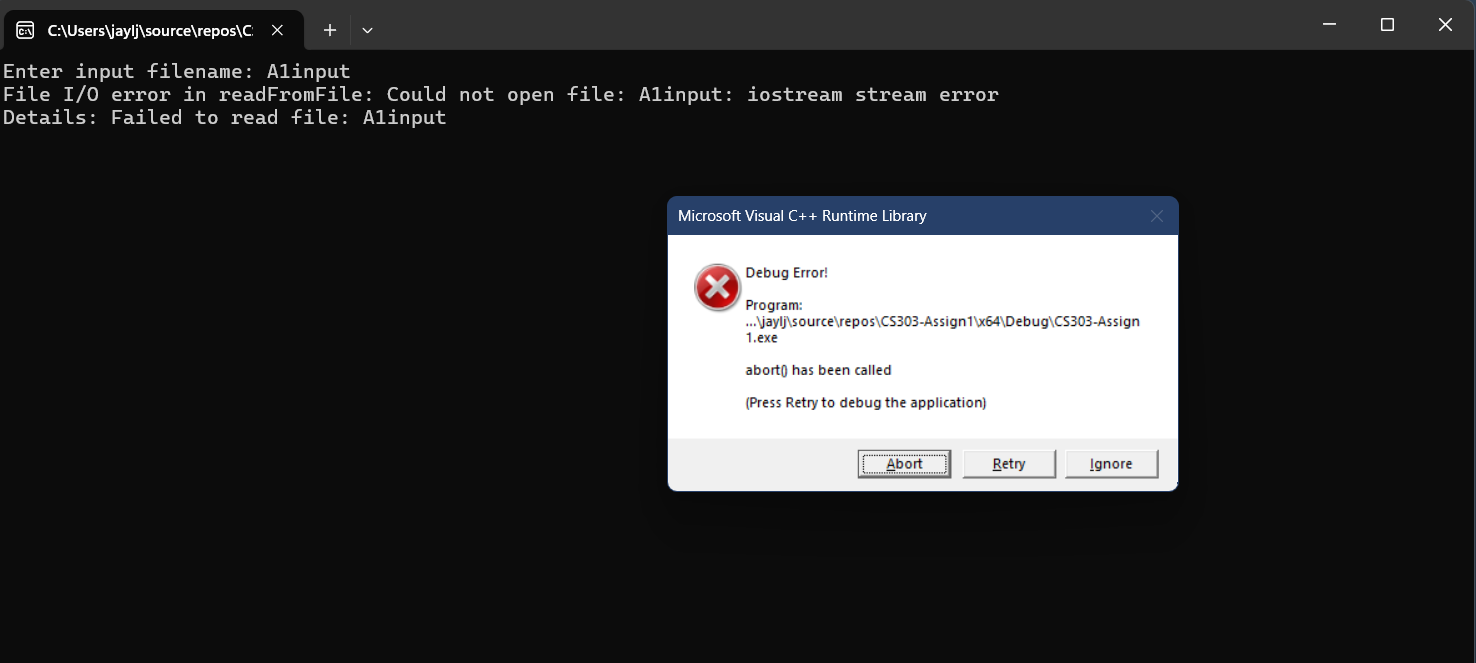
Why? Because it starts at the offsets of where the array starts in the memory.

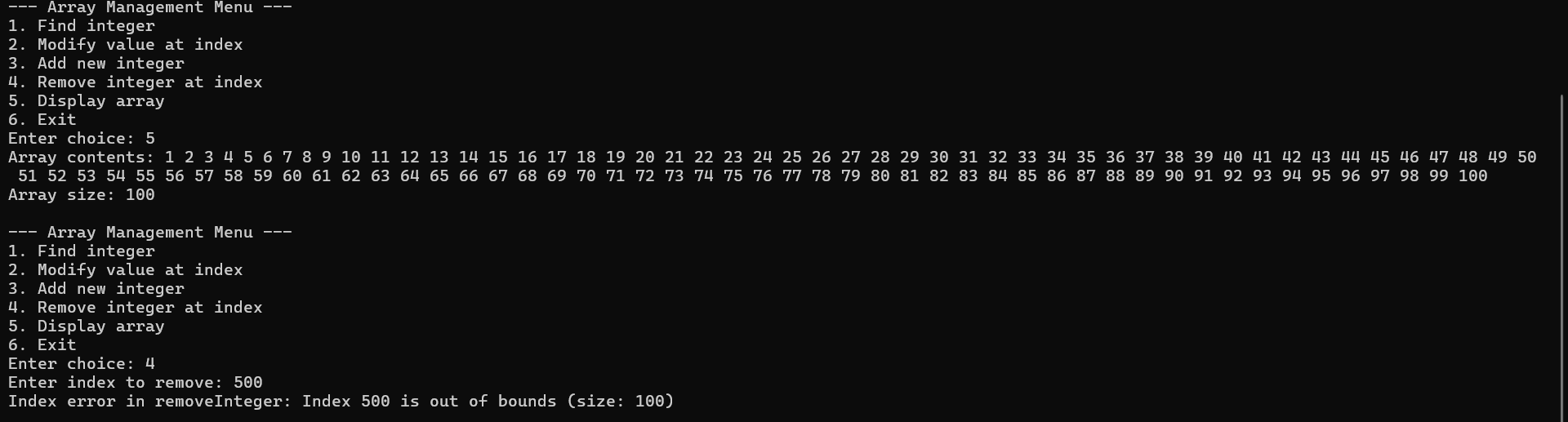
Basically, the program is starting at zero for each number place instead of 1.

  
  
You’ll be given six options: find the integers, let you find a certain number that you are looking for, modify values, let you update values within the array or return an old value back. Remove integer lets you remove an integer, display array will display the array itself, and exit will just end the program and let the user close off the program.  
  
Here are some photos of what each option does:  
  
Choice 1:

  
  
Choice one lets the user find a number within the array, letting the user know where the number is at/know the number’s index place.

Choice 2 (and 5):  


The reason why I have chosen two and five together is so you can understand things better. The user with choice two can modify the array and replace the old number with a new one within its place in the index, so when I changed index five’s value from six to two, two took the spot for index five. The user can go back and return the old one by using option 2 again. Choice five, just display the array, so the user can see all the changes and stuff.   
  
Choice 3:  
  
  
  
The user could add a new value within the array, and the program will automatically resize itself to have that extra space for any future addition of values.  
  
  
  
Choice 4:  
  
This just allows the user to remove a value from the array. Nothing else special.  
  
Choice 6:  
The user ends the program completely without having to press the red X in the top right corner.  
  
  
  
Now, there are some try & catch functions to catch errors. There is one piece of code that will catch an error that isn’t known, but here are some errors that have a proper try & catch:  
  
Memory error(I killed my laptop for this, don’t worry, it is fine):  
  
  
For me to even get this error, I was forced to use the nuclear option.  
So, if the user somehow used up all the memory, the program will catch the error and will let the user abort the program, trying to minimize the risk of damage to the computer or any device that the user is using. Even though it is unlikely as it takes so many entries. This exists in case.  
  
  
  
Failed input error:  
  
If the user puts the wrong input filename, the program will catch the error and let the user know without it bricking the program completely, where the user cannot do anything.

Out of Bound Error:  
  
This will happen when the user tries to do anything like removing a value that is beyond the array capacity. The program will catch the error, let the user know, and let them try again without giving the option of needing to abort completely.