No Debug For Lab G

Lab G

Background

Arrays can cause all sorts of problems, especially when you exceed the bounds of the array. Due to zero-based indexing (where the first element starts with index 0), it may confuse some people when they have to differentiate declaring or using an array. Now, however, that you know classes and other interesting C++ constructs, you will be able to make it less confusing for the user.

Problem

You will be creating a simplified array class that stores strings inside of an array. However, your class will have logic to allow for one-based indexing, whereas the first element is 1, not 0. This is to help those programmers just being introduced to arrays have an easier time.

Your class will ensure that the programmer cannot go outside the bounds of the array, ever, and that maps a "list" index to the array index (1 -> 0, 2 -> 1, 3 -> 2).

SimpleArray Class

When defining your member functions, remember that all index parameters will be 1 - MAX_ARRAY_SIZE, yet your array will be from 0 - MAX_ARRAY_SIZE-1. You must translate and check the array bounds in your member functions.

Create a class with three members:

1. a private array of strings with a maximum size of 30 (use a constant for this value)
2. a public member function that returns a bool and takes an integer and a string
   - This member function will set the integer element to the corresponding string value
   - This member function returns **true** only if it was able to set an element in the array to the given value
   - You must check the lower and upper bounds of the element index to ensure it is within the array's bounds
3. a public member function that returns a string and takes an integer
   - This member function will look up the given element and return it if it is within the bounds of the array
   - If the given integer is not within the boundaries of the private member array, return the string "ERROR"

Non-member functions

1. Create a **menu** function that prints the following menu:

```
s index value - Set a value in the array
g index - Get a value in the array
q - Quit
```

In this menu, the user will supply a command: s, g, or q. Your menu function will then check to see whether the command is s, g, or q. If it is not, it will repeat the menu until the command is s, g, or q. Your menu function

WILL NOT input the index or value for any of the commands, this will be done inside of your int main() function. When the command is deemed correct, your menu function will return s, g, or q back to int main().

2. Create a function that will initialize all of the array values to "EMPTY". This function will take no parameters, but will return an object of your class containing initialized "EMPTY" strings. Notice that this is NOT a member function, so you must use your getters or setters to set all of the values to "EMPTY".

Process

1. You will need to declare a SimpleArray class
2. Initialize the declared array using your non-member function
3. Just like Lab F, present a menu and get s, g, q
4. If the user specifies s (set): get an index and a value from the user and use them as parameters to your set function
5. If the user specifies g (get): get an index from the user and use it as a parameter to your get function
6. If the user specifies q (quit): quit your program
7. See "Outputs" for appropriate outputs
8. Repeat the process from step 3 until the user specifies q (quit).

Outputs

1. Setting a value (with s index value)
    1. If the value couldn't be set (given by your member function returning false)
    2. Output "Unable to set index X". Where X is the index that the user supplied.
    3. Otherwise, set the given index to the given value and do not output anything (just repeat the program).
2. Getting a value (with g index)
    1. If the value couldn't be retrieved (given by your member function returning "ERROR")
    2. Output "Unable to get index X". Where X is the index that the user supplied.
    3. Otherwise, get the value of the given index and output: "Element X = Y". Where X is the index and Y is the string at that given element

Additional Requirements

1. Your member functions must check the array bounds
2. Your member functions must do all of the translation between one-based element indexing and array element indexing
3. You must use appropriate accessor/getter notation
4. You must use your initialization non-member function to initialize your simple array class array elements to "EMPTY"
5. You must use a constant to define the maximum size of the array (which is 30 elements)
6. There must be NO cout/cin statements within your member functions

C++ Topics Covered

Classes
Getter/setter notation
Functions
Returning objects in functions
Arrays

Textbook Chapters Covered

Chapter 10.1
Chapter 10.2