Robert Elsom

2/4/2019

Project 2 Document

Design:

* Animal class
  + Base class for all animal types
  + Contains getters and setters for age, cost, number of babies, food cost, and profits
  + Classes need to be virtual to pass to derived classes
  + Included virtual destructor for rest of classes
* Tiger class
  + Contains default constructor that sets age
  + Returns tiger specific cost, 1 baby, food cost, and profit
* Turtle class
  + Contains default constructor that sets age
  + Returns turtle specific cost, 1 baby, food cost, and profit
* Penguin class
  + Contains default constructor that sets age
  + Returns penguin specific cost, 1 baby, food cost, and profit
* Zoo class
  + Responsible for start menu and prompts
  + inputs to create new animals
  + Calls age functions for all animals and increases by one day
  + Calculates food cost and profits for each day by calling function of each animal
  + Updates bank roll based on food cost and profits
  + Creates random event
    - One animal gets sick and dies
    - Bonus based on number of tigers
    - One adult animal has a baby
  + Creates functions to double the size of an animal array if the array is close to being filled.
  + Function with prompt to buy new adult animal to add to zoo

Test Cases:

Starting Menu test cases

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case | Input Values | Driver Function | Expected Outcome | Observed Outcome |
| Function a not an unsiged integer | A, 1.5, -4 | ValidStr() | Display error, repeat options to user | Display error, repeat options to user |
| Integer not a valid option | 5 | ValidStr() | Display error, repeat options to user | Display error, repeat options to user |
| Integer a valid option | 1,2,3 | ValidInt() | Creates new animal corresponding to that number | Creates new animal corresponding to that number |

Buy Adult Animal Prompt test case

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case | Input Values | Driver Function | Expected Outcome | Observed Outcome |
| Input is not a char y or n | A, 1.5, -4, yn | ValidStr() | Display error, repeat options to user | Display error, repeat options to user |
| Input is a y or Y | Y, y | ValidStr() | Go to prompt to buy new animal | Go to prompt to buy new animal |
| Input is a n or N | N, n | validStr() | Go to continue prompt | Go to continue prompt |

Buy new adult animal test case

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case | Input Values | Driver Function | Expected Outcome | Observed Outcome |
| Function a not an unsiged integer | A, 1.5, -4 | ValidStr() | Display error, repeat options to user | Display error, repeat options to user |
| Integer not a valid option | 5 | ValidStr() | Display error, repeat options to user | Display error, repeat options to user |
| Integer a valid option | 1,2,3 | ValidInt() | Creates new adult animal corresponding to that number | Creates new adult animal corresponding to that number |

Continue prompt test case

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function a not an unsiged integer | A, 1.5, -4 | ValidStr() | Display error, repeat options to user | Display error, repeat options to user |
| Integer not a valid option | 5 | ValidStr() | Display error, repeat options to user | Display error, repeat options to user |
| Integer a valid option | 1 | Menu()  If choice == 1 || choice ==2 | Print repeating prompts and get inputs for functions | Print prompts and get inputs for functions |
| Integer second option | 2 | Exit() | Quit program | Quit program |
|  |  |  |  |  |

Reflection:

This project was very hard to find a memory leak for me. Valgrind was throwing an error in a different part of the program from where the error was, which required me to go back through the entire program step by step to find it. In the end, somehow I was trying to delete a pointer memory then reassign it, but reassigned it before deleting. I spent almost an entire day just trying to find the memory leaks due to it pointing to the wrong parts of the program.

Another mistake I made was not reading the instructions fully or forgetting about the zoo class to begin with, I had created a game class that I ended up redoing the design for once seeing that I needed to implement a zoo class instead. Another part of not reading the instructions fully is that I started with one class for all the animals, so it was a array of pointers pointing to the class objects, which was causing some problems with my other functions. That instruction, once read, made things a lot easier when I got to change it to 3 different arrays for each animal.

My original design also changed slightly, where I added a bool function to test the type of animal each object was. I figure there is an easier way to figure that out, but could not find anything online about how to tell what object type is for derived classes other than returning a value from functions in each derived class.

Other than the minor change in design and the memory leaks, everything followed the second plan after rereading the instructions. It was just a lot of time writing the code and more time creating the design just due to the fact that the program is larger than what we have previously written.