CMPS 1600 Introduction to Computer Science II – Spring 14

4/4/14

9. Homework

Programming portion due **Thursday 4/10/14** at 11:55pm on Blackboard.

Please submit one `.cpp` file on Blackboard.
In order to receive any credit for the programming portions, you are required to thoroughly comment and test your code.

1. **Bears and Fish (20 points)**
   (a) (5 points) In C++, create an `Animal` class which is extended by a `Bear` class and by a `Fish` class. `Animal` should provide a static counter to count animal objects, and it should have a (possibly static) method to return the counter; see the code provided on the "C and C++ slides, part IV". It should also contain a destructor in which the counter is decreased. Test the total animal count by creating and deleting objects.
   Override the counter variable and method in the `Bear` class and in the `Fish` class to count bears and fish separately, in addition to counting animals. You can use the scope resolution operator to access the different static counters.
   (b) (15 points) Now, write a C++ program that simulates an ecosystem of bears and fish in a river. The river should be modeled as an array that stores pointers to `Animal` objects, which can be `Bear` objects, `Fish` objects, or empty (`NULL`). The simulation should proceed in multiple rounds. In each round, bears and fish attempt to move into adjacent array cells or they can stay where they are; at random. If two fish are about to collide in the same cell, then they stay where the are, but they create a new instance of type `Fish`, which is placed in a random empty position in the array. If a bear and a fish collide, however, then the fish dies (i.e., it is deleted, and the overall animal counter decreases).
      - Create actual objects to model these types of animals, and provide a visualization of the array after each time step. It might help to explicitly store the type of each animal as an attribute in the classes.
      - Random numbers can be generated as follows: Add `#include <cstdlib>`. Then call `srand(time(NULL));` once to initialize a random seed. Now you can generate a random integer between 0 and $n-1$ by calling `rand()%n`.
      - You are free to make design choices yourselves. Please document your choices in the comments. Be careful with iterating over all the animals while deleting or adding some of them; it is probably better to add all the fish after a full iteration.
      - You can earn up to 10 extra credit points by structuring the program exceptionally well or by enhancing it with extra features. For example: Use an iterator in a meaningful way, run meaningful test cases and add a report on your findings, allow bears to multiply as well, add genders to the animals, have different multiplication rates for bears and fish, add strength values to bears and fish, allow bears to starve, etc.
      - Hints: (1) An array that stores pointers to animal objects can be declared as `Animal* river[riverSize];`
        (2) Would you then access a method for an individual animal with `river[i].someMethod()` or `river[i]->someMethod()` ?
        (3) A method to print the whole river may have a header similar to this: `void printRiver(Animal** river, int n)` – why?