1: Using One Class

CSCI 6626 / 4526 Spring 2018

1 Goals

- To build a simple class with all the usual parts.
- To use constructors and destructors.
- To attach your class to the predefined output operator.
- To make a code module comprised of matching .hpp and .cpp files.
- To build a main program that incorporates a unit-test that will test all parts of your class.
- To build the first part of the term project.

Please note: This assignment provides a lot of guidance and tells you the details of what I expect you to do. As the term goes on, less and less guidance will be provided in the assignments.

2 Dice and Games

Dice. Many games are played with a pair of 6-sided dice. However, dice come in other shapes: 4-sided, 12-sided, and 20-sided dice certainly exist. Also, some games use 1 die, others use more than 2 dice. In this program, you will create a Dice class that can have any number of 6-sided dice. The class members you need are described below.

3 Instructions

Parts of the Dice class. Implement all the usual parts of a class, that is:

- Declare a private variable to store *nDice*, the number of dice in the set, and an *int* pointer for a dynamically allocated array of *nDice* pseudo-random values.
- Define one constructor *Dice(int n)* with a default parameter providing two ways to construct dice. The constructor must store its parameter in the corresponding variable, *nDice* and allocate an array of *nDice* integers; there is no need to initialize those integers. Then call *srand()* to initialize the random number generator; use *time(NULL)* as a parameter for *srand()*.
- Declare a destructor that will free the dynamically allocated array.
- Declare *public ostream& print(ostream&)* function that will print the array of dice-values on one line, separated by single spaces. Do not include newlines as part of the output. Whether or not a newline is desirable is usually dependent on the context in which a print() function is used, and they are supplied by the method that calls print().
- Outside the class but inside the .hpp file, declare a method for the output operator,
  inline ostream& operator <<(ostream&, Dice&)
  It must call your *print()* function with the appropriate parameter and return the ostream as the result.
- Define a function *public const int* roll()*. Fill the dice-value array with *nDice* random values. Return the array of dice-values as a *const int**. Because of the *const*, the calling program will have read-only access to the array; changing it will not be possible.

4 Testing

It is important to test every part of this program, and to cause every possible error comment to appear in your output. Create a test plan – that is, a list of cases that must be tested. For each case, say what input you need and what output or program behavior should be produced. Incorporate this test plan in a function called `unitDice()`, which will be called from `main()`.

Test your program with different values of nDice. Generate enough random values to demonstrate that every value in the range 1..6 can be produced by your dice. Open a text file in append mode and check for opening errors. Send your test output from all the test runs to this one file.

Due: February 6 Put copies of your test plan and your output in the same folder as your source code. The name of the folder should include both the problem number and your name. (Example: P1-Fischer). Zip up your directory and email it to my home.

5 Advice

Email for help promptly if you have any trouble of any sort. PLEASE follow the instructions. This is a short, easy program. Don't complicate it. The highest grades will be for the simplest programs that follow the specifications.