“Homework” from the Monday, October 30th, 2017 5024 Programming Meeting

Thinks of something you’d like to do, involving taking one or more numbers, performing some calculation, then returning a single number. Some suggestions, if you’re stuck:

* Look up the calculation for “compound interest”
* Look up the calculation for calculating the result of a “quadratic forumula”: f(x) = ax2 + bx + c
* Look up the calculation for “projectile motion”, specifically: how far something will fly, like a cannon ball, when shot at a certain angle at a certain speed.
* Something else that involves taking at least one number and returning a single number.

Make a function that performs this calculation. Remember:

* The main() function is in one file, but the function is in a *separate* file.
* Place the “function signature” into *another* header (.h) file. Remember the “header guards”.
* #include the header in the file that has the main() function.
* In the main function, ask the user for the numbers you need (using cin), “call the function”, passing these numbers into the function as “parameters”, and receiving the result into another variable.
* Print this result variable out to the screen (using cout)

Now make a class that will contain the function you made above, as a “method”:

* Decide on a class name that describes what the function does. Like something more abstract than what you are doing. For example, if you used the “quadratic formula” example, you could call your class cMathStuff.
* Remember to put the “c” (or “C”) in front of the name, so not “class Nijna”, but “class **c**Ninja” or “class **C**Ninja”.
  + This is convention, not required, but is quite common (and there’s a reason we do it).
  + The word “class” is in all *lower* case. So not “Class” or any other variation
  + Remember to put a semicolon after the closing brace (curly bracket):

**class cMySexyClass  
{  
};**

🡨 don’t forget the semicolon!

* + Make sure you use the word “public:” (all lower case) *inside* the braces (curly brackets) and *above* the function signature (which is now called a “method declaration” BTW).

**class cMySexyClass  
{  
public:  
 int myAmazingFunction(int x, int y, int z);   
  
};**

* Make a header (.h) file with the same name as the class (again, you *don’t* have to do this, like in Java, but it’s a very common convention. We will soon see classes inside files that don’t have the same name, though.).
  + Remember the header guards (just like with the function header you did earlier)
* Place the function signature inside the class, making it a “method” of that class.
* Make a .cpp file (an “implementation” file) which:
* #includes the .h file with the class definition
* Has the function with the classes’ scope added (so cNinja:: or cMathStuff:: or whatever RIGHT in front of the function name and AFTER the return value).
* #include the class header file in the file with the main() function
* “Create an instance of your class” (i.e. declare it like a variable: **cMySexyClass bob;**)
* “Call the method” on the class by using the class “instance” name (bob would be the “instance of the class” in the example above), by using the “dot operator” (aka a period or decimal symbol), followed by the method:  
    
  **int result = *bob.*myAmazingFunction( 1, 2, 3 );**
* Note that you *can still call* your original method, even if it’s got the *identical* name. For example, if your original function was called myAmazingFunction, like the method above *and* you still #included the header file that has this function signature, calling the function (instead of the method) happens when you don’t use the class instance (aka the “bob” variable), so you could (and often do) do things like this, where you call the method AND the function:  
    
  **int result1 = myAmazingFunction(1,2,3);  
    
  int result2 = bob.myAmazingFunction(4,5,6);**

🡨 Calls the method (of the class)

🡨 Calls the function