Coding Assignment 2: The PhysVec Class

The PhysVec class is short for Physics Vector. Physics vectors are used to describe things like velocity and acceleration - physics. They differ from ordinary vectors because I say so. You will take the class header file I am giving you and you will be creating the class implementation file (.cpp) that makes everything work. You will also create a main file that will be used for testing the class. I am providing you with a sample main file that I used when I was writing the class myself. I will repeat this later for emphasis!

There are several things that I am going to require of you.

**Basics:**

1. You will write and test your code on the professional level IDE - cLion

2. You will submit your results on TRACS. Do NOT put them in a zip file, just post the main, both class files and CMAKE file.

3. You will name the class PhysVec and the main file main. You will use the exact .h file I gave you.

4. Please treat this as if you were doing it on the job and you were going to provide this to your coworkers. If you don't understand what I mean by that, you better get it clarified.

5. I cannot and will not tell you exactly how to do this assignment. You are professionals in training; you need to deal with some ambiguity and figure things out. That's real life. You can always ask for clarification.

6. I will describe for each item in the .h file how I would attack it. Combined with the lecture material and googling and the reading, you should be able to do this assignment reasonably easily. If I have left something out, email me to ask and I can then provide an answer to everyone if appropriate.

7. You will need to follow the basic coding standards we will talk about in class. Do NOT use " using namespace std", use my naming conventions, good variable names, good comments, ect.

8. You will test your code with a variety of inputs that reflect limits and combinations and bad data reaction.

9. I will likely think of more requirements, possibly after you have already started so don't make code that's hard to update and don't get pissy if I do since this happens all the time to coders in the process of a real job.

**PhysVec Class Header File - PhysVec.h (see file)**

1. PhysVec();

This is the default constructor taking no arguments. It just creates a PhysVec instance. Simplest possible implementation.

*// Constructor that takes a vector to be turned into a PhysVec*2. PhysVec(**const** std::vector<**int**> & input\_vec);

This is the constructor for taking a vector<int> and making it a PhysVec. Our project spec limits the type to int. Recognize this is not unsigned int. You will need to set the instances a\_vec\_ variable to be the input\_vec.

*// overload the copy constructor*3. PhysVec::PhysVec(**const** PhysVec & rhs};

Just what it says. A copy constructor is used to generate a NEW INSTANCE meaning that a copy of the "rhs" instance does not already exist. The new instance has the "this" pointer so use it to set the new instances *"a\_vec\_"* variable to the "a\_vec\_ vector<int>" in "rhs". This can be done in one line of code.

*// overload the copy assignment operator*4. PhysVec &**operator**= (**const** PhysVec & rhs);

More complicated since the two instances already exist and the "rhs" might be a complex entity. This implementation will require two vector<int> iterators. One a const\_iterator and one not. Why? What is const and what is not?

One iterator is used for the "lhs and one for "rhs" to do the vector element by element copy. Notice that you are returning an object reference since we started with a pointer to an instance. (that's what the instance name refers to - a pointer) so when you return, you will be returning what? An instance pointer. (I will give you a hint "this pointer" :) )

5. PhysVec **operator**+ (**const** PhysVec & rhs) **const** ;

This is the high point of the PhysVec class. A PhysVes adds the same terms together in two vectors generating a third vector. ai + bi = ci for i = 1 to i= n.

This will require three vector iterators, two const\_iterators and one non const. You figure out which is which. Based on the function signature you should now realize that you will use the "this" pointer and the rhs reference. You need to make sure that both vectors are the same length and throw an invalid argument exception with an explanation if not. The tricky part is the vector c. It will be a local, temp vector and you will need to set it to the size of the other two. What is the vector method to do this? Watch out for <vector>.size. It returns a uint\_t type not an int. When you add arguments remember that an iterator is a pointer to a vector argument so dereferencing the iterator returns the argument! Also, you are returning the PhysVec vector so that the = operator has the right type to work with. The = operator works with two OBJECTS not OBJECT REFERENCES! Watch the const's.

6. **int** calculateDotProduct(**const** PhysVec & rhs) **const**;

<https://www.mathsisfun.com/algebra/vectors-dot-product.html>

Look at this link to figure out dot product if you cannot remember it. The dot product implementation will look a lot like the + operator except the function will be different obviously and since are returning an int, you will only need two iterators. You figure out which type.

7. **void** setPhysVec(**const** std::vector<**int**> & input\_vec);

This is a setter for a PhysVec. It is used with the default constructor which creates a non const PhysVec. If you are trying to change a const PhysVec you may need to cast the PhysVec as non const to use this function on it. You can always overload this function and make a second non-const taking version too.

Read this:

<https://www.cprogramming.com/tutorial/const_correctness.html>

This function should be easy by now since you already did the constructor and understand the "this" pointer. Hint Hint ( I am making this way too easy :( )

*// getters for a PhysVec - turns a PhysVec into a normal vec*

8. std::vector<**int**> &get\_a\_vec();

I refuse to lower myself to explain this.

9. **void** printvecs() **const** ;

To print a vector you need to use an iterator to walk through the elements. Enough said.

Submission:

You will need to write the implementation .cpp file. You will need to write a main file. Both of these will #include the PhysVec.h file. Because this is your first coding assignment with me, I am giving you the main file I used to test my work. This will likely not happen again. You need to do a much nicer one that has nice comments and tests all the cases since I left at least one or two out so you have to think about it. I will test your code with another main that my grader and I whip up. He's pretty smart so you might want to consider limits, combinations, error conditions, ect.

I want the three files plus your CMakeLists.txt file on TRACS. DO NOT BE LATE!

One last thing: If I didn't do something correctly that doesn't mean you can too. You find it, you fix it. I am pretty good at this but you never know, sometimes I put errors in on purpose :)

Oh Yeah, One last thing. If you think there is anything missing, you might want to ask in class.

Greg