1 Introduction

This is an exercise in object-oriented programming and operator overloading. This exercise covers the following topics:

- classes and class design
- constant functions and parameters
- arithmetic operators
- assignment operators
- member operators
- free operators
- stream operators
- error handling for streams
- reading from streams

2 Vector

In this exercise you are to create a class `Vector` that represents a 2-dimensional (mathematical) vector. You are to represent the vector as two floating point numbers (`double`), one for each component. It should be possible to

- initialize the vector,
- negate the vector,
- add and subtract vectors from each other,
- multipliy a vector with a constant (both from the left and the right),
- divide a vector with a constant,
- print a vector to arbitrary streams,
- read a vector from a stream,
- calculate the *dot product* of two vectors,
- compare vectors for equality and inequality,
- get the length of the vector,
- do any other operation you find necessary.

Make sure that you test everything! There are some testcases given in `main.cc`, but you should probably add more cases.

3 Operations

If you are unsure how the operations should work or if you don't know about vectors in mathematics, read the background section. If you feel comfortable with vectors and their operations you can keep on reading.

You are to implement the operations with operator overloading. All operators that mathematically *commutes* (i.e. all operators where the order doesn't matter) should commute for your vector class. For example: `v + w` should produce the same result as `w + v` (where `v` and `w` are of type `Vector`).

It should be possible to do *compound assignment* for each possible arithmetic operator. For example; it should be possible to do `v += w` or `v *= 0.5` etc.

You should avoid code duplication as much as possible. Prefer to call already implemented operators instead of reimplementing the same logic in each version of that operator. With this in mind i recommend that you first implement the compound assignment operators and then use them to build all other arithmetic operators.

The vector should be printable to *any* stream; this is done by declaring the first parameter of `operator<<` as a `std::ostream&`. The vector should be printed on the following format: `(x, y)`.

Example:

```
Vector v { 3.0, 4.0 };
std::cout << v << std::endl;
```

should produce `(3, 4)`.

Likewise should reading from a stream work for *any* stream, which is done by declaring the first parameter of `operator>>` as a `std::istream&`.

A vector should be read in the same format as the printing, but it should ignore any extra whitespace.

If any of the parenthesis are missing, or if the comma is missing, an error should be reported. This is done by setting the so called `failbit` of the stream. This failbit is set by calling the `setstate` function.

Example: `std::cin.setstate(std::ios::failbit);`

Note that a failed reading should **NOT** update the object you are reading to. The object should only be fully read or not read at all.

In this exercise you should think carefully about wheter or not an operator should be declared as a member function of `Vector` or if it should be declared outside the class. There is a big difference between the two; some operators cannot be declared inside the class at all!

Hint: Look in the header `<cmath>` for mathematics related functions.

4 Background

A 2-dimensional vector is a pair of two values, often denoted as $\begin{pmatrix}1\\2\end{pmatrix}$ or $\begin{pmatrix}1 & 2\end{pmatrix}$. These values are called *components*, often called x and y.

Vectors can be added together by simply adding each component individually, as such:

$$\begin{pmatrix}1\\2\end{pmatrix} + \begin{pmatrix}3\\4\end{pmatrix} = \begin{pmatrix}1+3\\2+4\end{pmatrix} = \begin{pmatrix}4\\6\end{pmatrix}$$

So adding two vectors produces a new vector. It works the same way for subtraction.

Vectors can be multiplied with constant values which means multiplying each component with said value, example:

$$0.5 \cdot \begin{pmatrix}1\\2\end{pmatrix} = \begin{pmatrix}0.5 \cdot 1\\0.5 \cdot 2\end{pmatrix} = \begin{pmatrix}0.5\\1\end{pmatrix}$$

This is the same from both *left* and *right*, that is $0.5 \cdot \begin{pmatrix}1\\2\end{pmatrix} = \begin{pmatrix}1\\2\end{pmatrix} \cdot 0.5$.

Division works the same way as multiplication, but it only works from the right, so $\begin{pmatrix}1\\2\end{pmatrix} /2 = \begin{pmatrix}1/2\\2/2\end{pmatrix} = \begin{pmatrix}0.5\\1\end{pmatrix}$ is valid, while $2/ \begin{pmatrix}1\\2\end{pmatrix}$ is not.

Note: it is the same as multiplying with the constant $\frac{1}{2}$.

The last operation you need to know for this exercise is the *dot product*, which is an operation that takes two vectors and produces a single value. It works by taking the product of each component and then sum them together. Example:

$$\begin{pmatrix}1\\2\end{pmatrix} \cdot \begin{pmatrix}3\\4\end{pmatrix} = 1 \cdot 3 + 2 \cdot 4 = 3 + 8 = 11.$$

You can get the *length* of a vector by applying Pythagoras theorem to the vector. The length of a vector is often denoted as $\left|\begin{pmatrix}1\\2\end{pmatrix}\right|$. Example:

$$\left|\begin{pmatrix}3\\4\end{pmatrix}\right| = \sqrt{3^2 + 4^2} = \sqrt{9 + 16} = \sqrt{25} = 5.$$