InFoMM C++ Skills Training

Practical 2 (Lectures 4-6)

1. Write code that sends the address of an integer to a function that prints out the value of the integer. Change the value of the integer and verify that the original integer is updated outside your function.
2. Write a function that accepts two floating point numbers, and swaps the values of these numbers.
   (a) Write this function using pointers
   (b) Write this function using references
3. Write a function that returns the scalar (dot) product of two `std::vector<double>` vectors.
4. The p-norm of a vector $\mathbf{v}$ of length n is given by

$$\|\mathbf{v}\|_p = \left(\sum_{i=1}^{n} |v_i|^p \right)^{1/p}$$

where p is a positive integer. Write a function to calculate the p-norm of a given `std::vector<double>`, where p takes the default value 2. Loop over the vector using (a) an index-based loop, (b) a range-based loop, (c) an iterator-based loop, and (d) the `std::accumulate` STL algorithm

5. Overload the function written in Q4 for the particular case of a `std::array<double,3>`. Now template your function to enable it to take and compile-time length N, i.e. `std::array<double,N>`

Hint: the definition of `std::array` is

```cpp
template< class T, std::size_t N> struct array;
```

6. Write a function multiply that may be used to multiply two matrices, given their sizes. You are free to choose any type to represent your matrices, but you might want to try either a `std::vector<double>` or a `std::vector<std::vector<double>>`

7. Implement the same matrix multiply in Eigen (http://eigen.tuxfamily.org) and time how long your function takes compared with Eigen for a relatively large matrix. (advanced) Try to improve the speed of your function as much as you can in the allotted time.