

C++ code 0.1.3: Casting the scalar type of a matrix

```
2 double casting() {
3     // Because the syntax Eigen::Matrix<type, n_rows, n_cols> is very
4     // cumbersome, Eigen provides convenience classes that work as shorthands.
5     // For example, Eigen::Matrix2d is shorthand for Eigen::Matrix<double, 2, 2>.
6
7     // Vectors are just a special type of matrices that have only one column.
8     // Thus, VectorXi is a shorthand for Eigen::Matrix<int, Eigen::Dynamic, 1>.
9     // Constant(n,x) creates a vector of size n and initializes the entries with
10    // the value x.
11    Eigen::VectorXi u = Eigen::VectorXi::Constant(2, 2);
12
13    // std::complex is a template class for complex numbers.
14    // Here we declare two complex numbers, with real and imaginary parts
15    // represented by doubles. z0 = 1 - i z1 = 5 + i
16    std::complex<double> z0(1, -1), z1(5, 1);
17
18    // Declare and initialize a size 2 vector of std::complex<double>.
19    Eigen::Vector2cd v;
20    v(0) = z0;
21    v(1) = z1;
22
23    double x = 0.0;
24    // TODO: (0-1.d) Calculate the inner product of u and v, and store the result
25    // in x. Hint: The inner product of two vectors is given by u.dot(v), but
26    // first we need to cast the "int" vector u to a "std::complex<double>"
27    // vector. Use u.cast<NEW TYPE>() to achieve this. The result of the inner
28    // product will be 1*(1-i) + 1*(5+i) = 6 + 0i, a real number. You can get the
29    // real part of an std::complex<double> using the method "real()".
30    // START
31    std::complex<double> z = u.cast<std::complex<double>>().dot(v);
32    x = z.real();
33    // END
34    return x;
35 }
```

Get it on  GitLab (MatrixClass.hpp).