Good things to know

- C++ is a case sensitive language (variable1 $\neq$ Variable1 $\neq$ vArIaBlE1)
- Remember the semicolon at the end of the line
- C++ is zero indexed, meaning containers start at index 0
- Compile time - when you compile your source code
- Runtime - while running your program, which is after compile time
- A good reference for the Eigen library is: http://eigen.tuxfamily.org/dox/AsciiQuickReference.txt

Suggested Coding Practice

- Write self documenting code
- Use very detailed variable names
- Have variable names start with lower case letters

Data Types

standard template library

```
unsigned int name1 = 1;             // integers >=0
int          name2 = 1;             // any integer
float        name3 = 3.14;          // floating pt number
double       name4 = 3.14;          // floating pt number with greater precision
string       name5 = "I like math"; // a string of characters
```

Data Structures

Both array and vector use the bracket operator (e.g [ ]) to access elements

array (static container)

An array is a fixed sized data structure. Once you define the array, it cannot be resized. This size must be specified at compile time. How to define an array:

```
Method1:
array<ObjectType,N_Objects> anyNameYouWant;

Method2:
array<ObjectType, N_Objects> anyNameYouWant = {{Object1, Object2 ..., Object_N}};
```

Compilable Code

Generic Example Code (Not Compilable)

Example of how to interact with an array:

```
array<double,4> randomArrrayName;
randomArrayName[0] = 0;
randomArrayName[1] = randomArrayName[0] + 1;
randomArrayName[2] = randomArrayName[1] + 2;
double iWantToAddTwoNumbersFromArray = randomArrayName[0] + randomArrayName[1];
// but what now is the value of randomArrayName[3]?
```

Example that will give you errors:

```
array<double,4> randomArrayName;
randomArrayName[4] = randomArrayName[5] + 2;
```

why does this not work? because the array was created with a length of 4 (0,1,2,3)

vector (dynamic container)

A vector is a variable sized data structure. It's initial size is 0. It's size can be changed during runtime. How to define a vector:

```
vector<ObjectType>
```

Example of how to interact with a vector:

```
vector<double> vectorOfDoubles;
vectorOfDoubles.push_back(1);
vectorOfDoubles[0] = 2; // this reassigns the value of the first entry in the vector
```

Example that will give you errors:

```
vector<int> vectorOfInts;
vectorOfInts[0] = 1;
```

why does this not work? because the vector has size 0 and you are trying to assign 1 to the first entry of the vector, which doesn't exist.

Eigen library

Provides static and dynamic vectors and matrices depending on how you define them.
Uses the parenthesis operator (e.g. ());
Static Vectors and Matrices

Compilable Code

Generic Example Code (Not Compilable)

```
Matrix<_standardDataType_,numRows,numCols> staticMatrix;
Matrix<_standardDataType_,numRows,1> effectiveStaticVector;
```

Dynamic Vectors and Matrices

```
unsigned int numRows = 10;
unsigned int numCols = 10;

//Method 1 for matrices:
MatrixXd dynamicMatrix1;
dynamicMatrix1.resize(numRows,numCols);
dynamicMatrix1.fill(0);

//Method 2 for matrices:
MatrixXd dynamicMatrix2(numRows,numCols);
dynamicMatrix2.fill(0);

//Method 1 for Vectors:
VectorXd dynamicVector1;
dynamicVector1.resize(numRows);
dynamicVector1.fill(0);

//Method 2 for Vectors:
VectorXd dynamicVector2(numRows);
dynamicVector2.fill(0);
```

Example accessing parts of a matrix:

```
MatrixXd dynamicMatrix(10,10);
dynamicMatrix.fill(0);
dynamicMatrix(0,0) = 1;
dynamicMatrix(3,2) = dynamicMatrix(0,0);
```

What can I ask/do with the Eigen Matrices and Vectors?

```
MatrixXd generalMatrix(10,10);
generalMatrix.fill(0);
unsigned int matrixRows = generalMatrix.rows(); // ask how many rows the matrix has
unsigned int matrixCols = generalMatrix.cols(); // ask how many columns the matrix has
MatrixXd transposedMatrix = generalMatrix.transpose(); // transpose the
// matrix and assign it to another matrix

// solve a linear system Ax = b
MatrixXd matrixA(10,10);
matrixA.fill(0); // always good practice to do this
```

Compilable Code

Generic Example Code (Not Compilable)

```
// fill matrix A with numbers ...
VectorXd vectorb(10);
vectorb.fill(0); // always good practice to do this
// fill vector b with numbers
VectorXd x = matrixA.lu().solve(vectorb);
```

Hello World Program

```
// -*- C++ -*-
#include "/ae108/Definitions.h"

int main(int arc, char *argv[]) {
  printf("Hello world!\n");
  return 0;
}
```

Adding the contents of two (stl) vectors into one sum

```
// -*- C++ -*-
#include "/ae108/Definitions.h"

int main(int arc, char *argv[]) {
   unsigned int numberOfVectorTerms = 10;
   vector<double> firstVectorOfDoubles;
   vector<double> secondVectorOfDoubles;

   // fill the vectors with numbers
   for (unsigned int index = 0; index < numberOfVectorTerms; index++){
      firstVectorOfDoubles[index] = index;
      secondVectorOfDoubles[index] = numberOfVectorTerms - index;
    }

    // add the numbers from each vector into one sum
    double sum = 0
    for (unsigned int index = 0; index < numberOfVectorTerms; index++){
       sum += firstVectorOfDoubles[index] + secondVectorOfDoubles[index];
    }
    return 0;
}
```

Compilable Code

Generic Example Code (Not Compilable)