Refactoring Code Examples

1 A Simple Class

The following class maps a n-dimensional real vector and its magnitude into C++.

Listing 1: **The MagVector Class**

```cpp
#ifndef _MAGVECTOR_H_
#define _MAGVECTOR_H_

#include <vector>
#include <iostream>
#include <cmath>
#include <stdexcept>

class MagVector
{
private:

  std::vector<float> _content;

public:

  MagVector(const int& _size):
    _content(_size,0.){
  };

  virtual ~MagVector(){};

  void print(){
    std::vector<float >::const_iterator vItr = _content.begin();
    std::vector<float >::const_iterator vEnd = _content.end();
    std::cout << "vector\t";
    for (; vItr!=vEnd; ++vItr){
      std::cout << *vItr << ", ";
    }
    std::cout << std::endl;
  }

  void append(const float& _new){
    _content.push_back(_new);
  }

  void set(const int& _index,const float& _new){
    if(_index>-1 && _index<_content.size())
      _content[_index] = (_new);
    else
      throw std::out_of_range("MagVector::set >> index out of range");
  }

  float get(const int& _index){
    try{
      return _content.at(_index);
    }
    catch(...){
      std::cerr << __FILE__<< ":\t" << "index out of range!\n";
      return 0.;
    }
  }

  float magnitude() const {
    float valueSquared = 0.;
    for (short index=0; index<_content.size(); ++index){
      valueSquared += ((_content[index])*(_content[index]));
    }

    if(valueSquared >= 0.)
      return std::sqrt(valueSquared);
    else
      return -1.;
  }

};

#endif /* _MAGVECTOR_H_ */
```

2 Simple Tests

Using boost, we write tests to ensure that all behavior and exceptions are covered.

Listing 2: **The MagVector Class Tests**

```
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE MagVectorTests
#include "MagVector.hh"

#include <boost/test/unit_test.hpp>

// BOOST_AUTO_TEST_CASE( testFails )
// {
//   BOOST_FAIL("This test fails!");
// }

BOOST_AUTO_TEST_SUITE( MagVectorSuite )

BOOST_AUTO_TEST_CASE( testValueGet )
{
  MagVector aVector(3);
  BOOST_CHECK_MESSAGE(aVector.get(0) == 0., "Content on 0 not on default value! ");
}

BOOST_AUTO_TEST_CASE( testValueGetAllDefaultZeros )
{
  MagVector aVector(3);
  BOOST_CHECK_MESSAGE(aVector.get(0) == 0.
                      && aVector.get(1) == 0.
                      && aVector.get(2) == 0.,
                      "Content on 0 not on default value! ");
}

BOOST_AUTO_TEST_CASE( testSetValueAtIndex )
{
  MagVector aVector(3);
  aVector.set(0,42);
  BOOST_CHECK_MESSAGE(aVector.get(0) == 42.,
                      "Content on 0 not at 42! ");
}

BOOST_AUTO_TEST_CASE( testSetValueAtOutOfRangeIndex )
{
  MagVector aVector(3);
  BOOST_CHECK_THROW(aVector.set(4,42), std::out_of_range );
}

BOOST_AUTO_TEST_CASE( testMagnitudeOf0 )
{
  MagVector aVector(3);
  BOOST_CHECK_MESSAGE(aVector.magnitude()==0., "vector magnitude not on default" );
}

BOOST_AUTO_TEST_CASE( testMagnitudeOf4 )
{
  MagVector aVector(3);
  aVector.set(1,4);
  BOOST_CHECK_MESSAGE(aVector.magnitude()==4, "vector magnitude unequal 4" );
}

BOOST_AUTO_TEST_SUITE_END()
```

3 Simple Tests on a Fixture

Often classes require input data that is complex to instantiate. Instead of setting up MagVector per test case, a class can do that automatically every time a test case is called (this class is dubbed a `TestFixture`).

Listing 3: **The MagVector Class Tests on a Fixture**

```cpp
#define BOOST_TEST_DYN_LINK
#define BOOST_TEST_MODULE MagVectorTestsSuiteAndFixture
#include "MagVector.hh"
#include <boost/test/unit_test.hpp>

class MagVectorFixture
{

public:

  MagVector ThreeVector;

  MagVectorFixture():
    ThreeVector(3)
  {
    BOOST_MESSAGE( "setup fixture" );
  };

  virtual ~MagVectorFixture()  { BOOST_MESSAGE( "teardown fixture" ); };


};

BOOST_FIXTURE_TEST_SUITE( MagVectorSuite, MagVectorFixture )

BOOST_AUTO_TEST_CASE( testValueGet )
{
  BOOST_CHECK_MESSAGE(ThreeVector.get(0) == 0., "Content on 0 not on default value! ");
}

BOOST_AUTO_TEST_CASE( testValueGetAllDefaultZeros )
{
  BOOST_CHECK_MESSAGE(ThreeVector.get(0) == 0.
                      && ThreeVector.get(1) == 0.
                      && ThreeVector.get(2) == 0.,
                      "Content on 0 not on default value! ");
}

BOOST_AUTO_TEST_CASE( testSetValueAtIndex )
{
  ThreeVector.set(0,42);
  BOOST_CHECK_MESSAGE(ThreeVector.get(0) == 42.,
                      "Content on 0 not at 42! ");
}

BOOST_AUTO_TEST_CASE( testSetValueAtOutOfRangeIndex )
{
  BOOST_CHECK_THROW(ThreeVector.set(4,42), std::out_of_range );
}

BOOST_AUTO_TEST_CASE( testMagnitudeOf0 )
{
  BOOST_CHECK_MESSAGE(ThreeVector.magnitude()==0., "vector magnitude not on default" );
}

BOOST_AUTO_TEST_CASE( testMagnitudeOf4 )
{
  ThreeVector.set(1,4);
  BOOST_CHECK_MESSAGE(ThreeVector.magnitude()==4, "vector magnitude unequal 4" );
}

BOOST_AUTO_TEST_SUITE_END()
```