# Lab 8

## Vector Example

**Introduction**

In this lab you will implement the Vector example from scratch. This will serve as a review of several topics in C++ programming. You will gain practice using a static data member, implementing a friend function, and overloading an operator.

**Suggested Time:** 45 minutes.

**Exercise 1.**

In this exercise you will examine the initial version of the Vector example. Review the following features:

- A static data member to specify the size of all vectors.
- A constructor to dynamically allocate an array of integers to hold the vector's data.
- A destructor to delete the array.
- A **Set()** method to initialize the data in the vector from an array.
- A **Print()** method to print out the vector's data on one line.
- A **DotProduct()** method that will obtain the dot product of another vector and itself, returning an integer.

1. Build and run. You should see this output (which will also be the same output for the next two exercises.) 1*1 + 2*4 + 3*9 = 36.

```
1   2   3

1   4   9

36
```

**Exercise 2**

In this exercise you will modify your program to make **DotProduct()** a friend function in place of a member function. This will also change how your test program invokes the function.

1. Modify the prototype of **DotProduct()** in the header file to be a friend.

2. Modify the implementation of **DotProduct()** in the file **Vector.cpp** to reflect that the function is now not a member.

3. Finally, modify the test program appropriately. Build and run.

**Exercise 3**

In the last exercise you will modify the definition of the DotProduct() function to be the overloaded multiplication operator.

1. In the header file replace **DotProduct** by **operator ***.

```
friend int operator * (const Vector& v1,
                       const Vector& v2);
```

2. Do the same in the implementation file.

```
int operator *(const Vector& v1,
               const Vector& v2)
```

3. Finally, in the test program replace the function call by infix notation using the operator you have defined.

```
cout << v1 * v2 << endl;
```

4. Build and run. You should get identical output to the previous versions.