ESS201: C++ Programming

Jaya Sreevalsan Nair *
International Institute of Information Technology, Bangalore

Term I: 2017-18 (Lab on 2017-10-16)

Tasks:

1. Rewrite `struct vector3d` from Lab01 to a class interface with (private) data members (`float _x, _y, _z`), in a single C++ file.

```
struct vector3d {
  float x, y, z ; // data
}
```

```
class vector3d {
 private:
   float _x, _y, _z ; // data
} ;
```

   - The constructor must take in three float values. Additionally, the class must have a default constructor, destructor, copy constructor, public members `get_x(), get_y(), get_z()` to access the private data members, and public member functions `set_x, set_y, set_z` to modify the private data members.
   - The class member changes to (set 1):

```
vector3d& add(vector3d b)
vector3d& subtract(vector3d b)
float& dot(vector3d b)
vector3d& cross(vector3d b)
float& norm()
```

     as opposed to the functions using struct (set 2):

```
vector3d add(vector3d a, vector3d b)
vector3d subtract(vector3d a, vector3d b)
float dot(vector3d a, vector3d b)
vector3d cross(vector3d a, vector3d b)
float norm(vector3d a)
```

     Set 1 can be assumed to be public members. One can assume that the `vector3d a` in all functions in set 2 are subsumed as the object itself in set 1. e.g. for a slight variant from the required member function:

```
float dot(vector3d a, vector3d b) {
   return (a.x*b.x+a.y*b.y+a.z*b.z) ;
```

     becomes:

```
float dot(vector3d b) {
   return (_x*b.get_x()+_y*b.get_y() + _z*b.get_z()) ;
```

   - The scalar-vector multiplication function, which is public, has to be overloaded:

```
vector3d scalar_product(float a) ;
vector3d scalar_product(int a) ;
```

*(jnair@iiitb.ac.in)

- A friend (ostream) operator[1] for printing the vector must be declared and defined in the vector3d class. The function declaration would be:
  `friend std::ostream& operator<<(std::ostream &os, vector3d &v) ;`

[1] Note: the vectors to be inputted and outputted are formatted with its three components separated by a single space.

- Input: should take in 4 lines of input at commandline prompt, i.e. first two lines for vectors, next one for float scalar, and the last one for int scalar. e.g.

```
3 3.4 5
2.1 4.2 4.1
0.5
3
```

- Output: should be in 8 lines, i.e. for addition of two input vectors; then for subtraction; then for scalar-vector multiplication of the float scalar value with the first vector; then for scalar-vector multiplication of the integer scalar value with the second vector; fifth line is for dot product of the two vectors; next one for the cross product; and the last two lines giving the L2 norm of each of the two vectors. e.g. output for the afore-mentioned input is:

```
5.1 7.6 9.1
0.9 -0.8 0.9
1.5 1.7 2.5
6.3 12.6 12.3
41.08
-7.06 -1.8 5.46
6.74981
6.23378
```