```
1  #pragma once
2  #include <math.h>
3
4  # define PI_2 6.283
5
6  class MyVector {
7  public:
8
9      // Constructors
10     MyVector() {
11         x = 0;
12         y = 0;
13         z = 0;
14     }
15     MyVector(float inputX, float inputY, float inputZ) {
16         x = inputX;
17         y = inputY;
18         z = inputZ;
19     };
20
21
22     bool operator==(const MyVector &Vector)const {
23         return x == Vector.x && y == Vector.y && z == Vector.z;
24     }
25
26     //MyVector arithmetic functions
27     MyVector operator+(const MyVector &Vector)const {
28         return MyVector(x + Vector.x, y + Vector.y, z + Vector.z);
29     }
30     MyVector operator-(const MyVector &Vector)const {
31         return MyVector(x - Vector.x, y - Vector.y, z - Vector.z);
32     }
33     MyVector operator*(const MyVector &Vector)const {
34         return MyVector(x * Vector.x, y * Vector.y, z * Vector.z);
35     }
36     MyVector operator/(const MyVector &Vector)const {
37         return MyVector(x / Vector.x, y / Vector.y, z / Vector.z);
38     }
39
40     // Overload for scalar functions
41     MyVector  operator+(float scalar)const {
42         return MyVector(x + scalar, y + scalar, z + scalar);
43     }
44     MyVector  operator-(float scalar)const {
45         return MyVector(x - scalar, y - scalar, z - scalar);
46     }
47     MyVector  operator*(float scalar)const {
48         return MyVector(x * scalar, y * scalar, z * scalar);
49     }
50     MyVector  operator/(float scalar)const {
51         return MyVector(x / scalar, y / scalar, z / scalar);
52     }
53
```

```
54
55      //MyVector cross(MyVector Vector);
56
57      float dotProduct(const MyVector &Vector)const {
58          return x * Vector.x + y * Vector.y + z * Vector.z;
59      }
60
61      float length() const {
62          return (float)sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2));
63      }
64
65      MyVector normalize()const {
66          float length = (float)sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2));
67          if (length == 0) {
68              return MyVector(0, 0, 0);
69          }
70
71          return MyVector(x / length, y / length, z / length);
72
73      }
74
75      float AngelBetweenVectors(const MyVector &Vector)const {
76          float LengthVector1 = (float)sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2));
77          float LengthVector2 = Vector.length();
78
79          if (LengthVector1 == 0 || LengthVector1 == 0) {
80              return 0;
81          }
82
83          return static_cast<float>(acos(dotProduct(Vector)) / (LengthVector1 *
               LengthVector2) * 360 / PI_2);
84      }
85
86      MyVector SetMaxLength(float MaxLength) const {
87          float l = length();
88          if (l > MaxLength) {
89              return normalize() * MaxLength;
90          }
91
92          return *this;
93      }
94
95      float x;
96      float y;
97      float z;
98  };
```