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

```
54
55
56      //MyVector cross(MyVector Vector);
57
58      float dotProduct(const MyVector &Vector)const {
59          return x*Vector.x+y*Vector.y+z*Vector.z;
60      }
61
62      float length() const {
63          return (float) sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2));
64      }
65
66      MyVector normalize()const {
67          float length = (float) sqrt(pow(x, 2) + pow(y, 2) + pow(z, 2));
68          if (length == 0) {
69              return MyVector(0, 0, 0);
70          }
71
72          return MyVector(x / length, y / length, z / length);
73
74      }
75
76      float AngelBetweenVectors(const MyVector &Vector)const {
77          float LengthVector1 = (float) sqrt(pow(x, 2) + pow(y, 2) + pow(z,
              2));
78          float LengthVector2 = Vector.length();
79
80          if (LengthVector1 == 0 || LengthVector1 == 0) {
81              return 0;
82          }
83
84          return static_cast<float>(acos(dotProduct(Vector)) / (LengthVector1 *
               LengthVector2) * 360 / PI_2);
85      }
86
87      MyVector SetMaxLength(float MaxLength) const {
88          float l = length();
89          if (l > MaxLength) {
90              return normalize() * MaxLength;
91          }
92
93          return *this;
94      }
95
96      float x;
97      float y;
98      float z;
99  };
```