```
using System;

namespace ToyBox {
    public partial struct RGB {
        // see Blender
        public static byte FloatToByte(float inValue) {
            return inValue<=0.0f ? (byte)0
                                 : ( inValue>1.0f-0.5f/255.0f ? (byte)255
                                                              : (byte)(255.0f*inValue+0.5f) );
        }

        static RGB sWhite=new RGB(1,1,1);
        static public RGB White {
            get { return sWhite; }
        }

        static public RGB One {
            get { return sWhite; }
        }

        static RGB sBlack=new RGB(0,0,0);
        static public RGB Black {
            get { return sBlack; }
        }

        static public RGB Zero {
            get { return sBlack; }
        }

        static RGB sRed=new RGB(1,0,0);
        static public RGB Red {
            get { return sRed; }
        }

        static RGB sGreen=new RGB(0,1,0);
        static public RGB Green {
            get { return sGreen; }
        }

        static RGB sBlue=new RGB(0,0,1);
        static public RGB Blue {
            get { return sBlue; }
        }

        static RGB sMagenta=new RGB(1,0,1);
        static public RGB Magenta {
            get { return sMagenta; }
        }

        static RGB sCyan=new RGB(0,1,1);
        static public RGB Cyan {
            get { return sCyan; }
        }

        public float R,G,B;

        public RGB(float inR,float inG,float inB) {
            R=inR;
            G=inG;
            B=inB;
        }

        public RGB(RGB inSrcColor) {
            R=inSrcColor.R;
            G=inSrcColor.G;
            B=inSrcColor.B;
        }

        public RGB(float inIntensity) {
            R=G=B=inIntensity;
        }

        public void Init(float inR,float inG,float inB) {
            R=inR;
            G=inG;
            B=inB;
        }

        //-------------------------------------------------------------
        // 演算子オーバーロード
        //-------------------------------------------------------------
        // 単項マイナス
        static public RGB operator-(RGB inColor) {
            return new RGB(-inColor.R,-inColor.G,-inColor.B);
        }

        static public RGB operator+(RGB inColor1,RGB inColor2) {
            float r=inColor1.R+inColor2.R;
            float g=inColor1.G+inColor2.G;
            float b=inColor1.B+inColor2.B;
            return new RGB(r,g,b);
        }

        static public RGB operator-(RGB inColor1,RGB inColor2) {
            float r=inColor1.R-inColor2.R;
            float g=inColor1.G-inColor2.G;
            float b=inColor1.B-inColor2.B;
            return new RGB(r,g,b);
        }

        static public RGB operator*(RGB inColor1,RGB inColor2) {
            float r=inColor1.R*inColor2.R;
            float g=inColor1.G*inColor2.G;
            float b=inColor1.B*inColor2.B;
            return new RGB(r,g,b);
        }

        static public RGB operator*(float inK,RGB inColor) {
            return new RGB(inK*inColor.R,inK*inColor.G,inK*inColor.B);
        }

        static public RGB operator*(RGB inColor,float inK) {
            return new RGB(inK*inColor.R,inK*inColor.G,inK*inColor.B);
        }

        static public RGB operator/(RGB inColor1,RGB inColor2) {
            float r=inColor1.R/inColor2.R;
            float g=inColor1.G/inColor2.G;
            float b=inColor1.B/inColor2.B;
            return new RGB(r,g,b);
        }

        static public RGB operator/(RGB inColor,float inK) {
            float invK=1.0f/inK;
            return new RGB(inColor.R*invK,inColor.G*invK,inColor.B*invK);
        }

        static public bool operator==(RGB inColor1,RGB inColor2) {
            return inColor1.R==inColor2.R && inColor1.G==inColor2.G && inColor1.B==inColor2.B;
        }

        static public bool operator!=(RGB inColor1,RGB inColor2) {
            return inColor1.R!=inColor2.R || inColor1.G!=inColor2.G || inColor1.B!=inColor2.B;
        }

        public override bool Equals(object inObj) {
            if(inObj==null || inObj.GetType()!=GetType()) {
                return false;
            }
            RGB rgb=(RGB)inObj;
            return this==rgb;
        }

        public override int GetHashCode() {
            return R.GetHashCode() ^ G.GetHashCode() ^ B.GetHashCode();
        }

        public RGB Clamp(float inMin=0,float inMax=1) {
            return new RGB(clamp(R,inMin,inMax),clamp(G,inMin,inMax),clamp(B,inMin,inMax));
        }
        float clamp(float inT,float inMin,float inMax) {
            if(inT<inMin) {
                return inMin;
            } else if(inMax<inT) {
                return inMax;
            } else {
                return inT;
            }
        }

        override public string ToString() {
            return "["+R+","+G+","+B+"]";
        }
    }
}
```