```
//#define PROPERTY_XYZ

using System;

namespace ToyBox {
    public class Vector3 {
        // 内積
        static public float Dot(Vector3 inV1,Vector3 inV2) {
            return inV1.X*inV2.X+inV1.Y*inV2.Y+inV1.Z*inV2.Z;
        }

        // 外積
        static public Vector3 Cross(Vector3 inV1,Vector3 inV2) {
            return new Vector3(inV1.Y*inV2.Z-inV1.Z*inV2.Y,
                               inV1.Z*inV2.X-inV1.X*inV2.Z,
                               inV1.X*inV2.Y-inV1.Y*inV2.X);
        }

        // 正規化したベクトルを返す。inV の値は変化しない。
        static public Vector3 UnitVector(Vector3 inV) {
            float s=inV.InvLength;
            return s*inV;
        }

        static Vector3 mZero=new Vector3(0,0,0);
        static public Vector3 Zero {
            get { return mZero; }
        }

        static Vector3 mXAxis=new Vector3(1,0,0);
        static public Vector3 XAxis {
            get { return mXAxis; }
        }

        static Vector3 mYAxis=new Vector3(0,1,0);
        static public Vector3 YAxis {
            get { return mYAxis; }
        }

        static Vector3 mZAxis=new Vector3(0,0,1);
        static public Vector3 ZAxis {
            get { return mZAxis; }
        }

#if !(PROPERTY_XYZ)
        public float X,Y,Z;
#else
        float mX,mY,mZ;
        public float X {
            set { mX=value; }
            get { return mX; }
        }
        public float Y {
            set { mY=value; }
            get { return mY; }
        }
        public float Z {
            set { mZ=value; }
            get { return mZ; }
        }
#endif

        public float Length {
            get { return (float)Math.Sqrt(X*X+Y*Y+Z*Z); }
        }

        public float InvLength {
            get { return (float)(1.0/Math.Sqrt(X*X+Y*Y+Z*Z)); }
        }

        public Vector3() {
            X=Y=Z=0;
        }
```

```
        public Vector3(float inX,float inY,float inZ) {
            X=inX;
            Y=inY;
            Z=inZ;
        }

        public Vector3(Vector3 inSrc) {
            X=inSrc.X;
            Y=inSrc.Y;
            Z=inSrc.Z;
        }

        public void Init(float inX,float inY,float inZ) {
            X=inX;
            Y=inY;
            Z=inZ;
        }

        public void InitWithNormalize(float inX,float inY,float inZ) {
            float invLen=(float)(1/Math.Sqrt(inX*inX+inY*inY+inZ*inZ));
            X=inX*invLen;
            Y=inY*invLen;
            Z=inZ*invLen;
        }

        // 自分自身を正規化する
        public void Normalize() {
            float invLength=InvLength;
            X*=invLength;
            Y*=invLength;
            Z*=invLength;
        }

        // 演算子オーバーロード等
        static public Vector3 operator+(Vector3 inVec) {
            return inVec;
        }

        static public Vector3 operator-(Vector3 inVec) {
            return new Vector3(-inVec.X,-inVec.Y,-inVec.Z);
        }

        static public bool operator==(Vector3 inV1,Vector3 inV2) {
            return inV1.X==inV2.X
                && inV1.Y==inV2.Y
                && inV1.Z==inV2.Z;
        }

        static public bool operator!=(Vector3 inV1,Vector3 inV2) {
            return !(inV1==inV2);
        }

        static public Vector3 operator*(float inS,Vector3 inV) {
            return new Vector3(inS*inV.X, inS*inV.Y, inS*inV.Z);
        }

        static public Vector3 operator*(Vector3 inV,float inS) {
            return inS*inV;
        }

        static public Vector3 operator/(Vector3 inV,float inS) {
            float invS=1.0f/inS;
            return new Vector3(inV.X*invS, inV.Y*invS, inV.Z*invS);
        }

        static public Vector3 operator+(Vector3 inV1,Vector3 inV2) {
            return new Vector3(inV1.X+inV2.X,
                               inV1.Y+inV2.Y,
                               inV1.Z+inV2.Z);
        }

        static public Vector3 operator-(Vector3 inV1,Vector3 inV2) {
            return new Vector3(inV1.X-inV2.X,
                               inV1.Y-inV2.Y,
```

```
                                inV1.Z-inV2.Z);
        }

        public override bool Equals(object inObj) {
            if(inObj==null
            || inObj.GetType()!=GetType()) {
                return false;
            }
            Vector3 vec3=(Vector3)inObj;
            return this==vec3;
        }

        public override int GetHashCode() {
            return X.GetHashCode()
                 ^ Y.GetHashCode()
                 ^ Z.GetHashCode();
        }

        override public string ToString() {
            return "("+X+","+Y+","+Z+")";
        }
    }
}
```