```
#pragma once

// 2次元ベクトルクラス
class Vector2D
{
public:
	float x;
	float y;

public:
	Vector2D() : x(0.0f), y(0.0f)
	{

	}
	Vector2D(float scalar) : x(scalar), y(scalar)
	{

	}
	Vector2D(float mx, float my) : x(mx), y(my)
	{

	}

public:
	// 演算子オーバーロード
	Vector2D& operator = (const Vector2D& location)
	{
		this->x = location.x;
		this->y = location.y;

		return *this;
	}

	const Vector2D operator + (const Vector2D& location)
	{
		float x = this->x + location.x;
		float y = this->y + location.y;

		return Vector2D(x, y);
```

```
}
const Vector2D operator - (const Vector2D& location)
{
	float x = this->x - location.x;
	float y = this->y - location.y;

	return Vector2D(x, y);
}
const Vector2D operator * (const float& scalar)
{
	float x = this->x * scalar;
	float y = this->y * scalar;

	return Vector2D(x, y);
}
const Vector2D operator * (const Vector2D& location)
{
	float x = this->x * location.x;
	float y = this->y * location.y;

	return Vector2D(x, y);
}
const Vector2D operator / (const float& scalar)
{
	if (scalar < 1e-6f)
	{
		return Vector2D(0.0f);
	}

	return Vector2D(this->x / scalar, this->y / scalar);
}
const Vector2D operator / (const Vector2D& location)
{
	if (location.x < 1e-6f)
	{
		return Vector2D(0.0f);
	}
	if (location.y < 1e-6f)
	{
```

```
            return Vector2D(0.0f);
      }

      return Vector2D(this->x / location.x, this->y / location.y);
}

Vector2D& operator += (const Vector2D& location)
{
      this->x += location.x;
      this->y += location.y;

      return *this;
}
Vector2D& operator -= (const Vector2D& location)
{
      this->x -= location.x;
      this->y -= location.y;

      return *this;
}
Vector2D& operator *= (const float& scalar)
{
      this->x *= scalar;
      this->y *= scalar;

      return *this;
}
Vector2D& operator *= (const Vector2D& location)
{
      this->x *= location.x;
      this->y *= location.y;

      return *this;
}
Vector2D& operator /= (const float& scalar)
{
      if (scalar < 1e-6f)
      {
            this->x = 0.0f;
```

```
                this->y = 0.0f;
        }
        else
        {
                this->x /= scalar;
                this->y /= scalar;
        }

        return *this;
    }
    Vector2D& operator /= (const Vector2D& location)
    {
        if (location.x < 1e-6f)
        {
                this->x = 0.0f;
                this->y = 0.0f;
        }
        else if (location.y < 1e-6f)
        {
                this->x = 0.0f;
                this->y = 0.0f;
        }
        else
        {
                this->x /= location.x;
                this->y /= location.y;
        }
        return *this;
    }
};
```