```cpp
#include "Matrix3x3.h"
#include <cassert>
#include <cmath>

Matrix3x3 Matrix3x3::operator*(const Matrix3x3& aOther) const
 noexcept {
    Matrix3x3 lResult(
        Vector3D(
            fRows[0].dot(aOther.column(0)),
            fRows[0].dot(aOther.column(1)),
            fRows[0].dot(aOther.column(2))
        ),
        Vector3D(
            fRows[1].dot(aOther.column(0)),
            fRows[1].dot(aOther.column(1)),
            fRows[1].dot(aOther.column(2))
        ),
        Vector3D(
            fRows[2].dot(aOther.column(0)),
            fRows[2].dot(aOther.column(1)),
            fRows[2].dot(aOther.column(2))
        )
    );
    return lResult;
}

std::ostream& operator<<(std::ostream& aStream, const Matrix3x3&
 aMatrix) {
    return aStream << "[" << aMatrix.fRows[0].toString() << ","
                   << aMatrix.fRows[1].toString() << ","
                   << aMatrix.fRows[2].toString() << "]";
}

float Matrix3x3::det() const noexcept {
    float lA1 = fRows[0].x();
    float lB1 = fRows[0].y();
    float lC1 = fRows[0].w();
    float lA2 = fRows[1].x();
    float lB2 = fRows[1].y();
    float lC2 = fRows[1].w();
    float lA3 = fRows[2].x();
    float lB3 = fRows[2].y();
    float lC3 = fRows[2].w();

    return lA1 * (lB2 * lC3 - lC2 * lB3) - lB1 * (lA2 * lC3 - lA3 *
     lC2) + lC1 * (lA2 * lB3 - lA3 * lB2);
}

Matrix3x3 Matrix3x3::transpose() const noexcept {
    return Matrix3x3(
        Vector3D(fRows[0].x(), fRows[1].x(), fRows[2].x()),
```

```
        Vector3D(fRows[0].y(), fRows[1].y(), fRows[2].y()),
        Vector3D(fRows[0].w(), fRows[1].w(), fRows[2].w())
    );
}

bool Matrix3x3::hasInverse() const noexcept {
    return det() != 0;
}


Matrix3x3 Matrix3x3::inverse() const noexcept {
    float lDet = this->det();
    assert(lDet != 0); //

    Matrix3x3 lInverse(
        Vector3D(
            (fRows[1].y() * fRows[2].w() - fRows[2].y() *
             fRows[1].w()),
            -(fRows[0].y() * fRows[2].w() - fRows[2].y() *
             fRows[0].w()),
            (fRows[0].y() * fRows[1].w() - fRows[1].y() *
             fRows[0].w())
        ),
        Vector3D(
            -(fRows[1].x() * fRows[2].w() - fRows[2].x() *
             fRows[1].w()),
            (fRows[0].x() * fRows[2].w() - fRows[2].x() *
             fRows[0].w()),
            -(fRows[0].x() * fRows[1].w() - fRows[1].x() *
             fRows[0].w())
        ),
        Vector3D(
            (fRows[1].x() * fRows[2].y() - fRows[2].x() *
             fRows[1].y()),
            -(fRows[0].x() * fRows[2].y() - fRows[2].x() *
             fRows[0].y()),
            (fRows[0].x() * fRows[1].y() - fRows[1].x() *
             fRows[0].y())
        )
    );

    return lInverse * (1.0f / lDet);
}
```