```cpp
#include "Matrix3x3_PS1.h"

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

std::ostream& operator<<(std::ostream& aOStream, const Matrix3x3& aMatrix) {
    size_t lIndex = 0;
    aOStream << "[";
    while (lIndex < 3) {
        aOStream << aMatrix.row(lIndex).toString();
        if(lIndex++ !=2) aOStream << ",";
    }
    aOStream << "]";
    return aOStream;
}


float Matrix3x3::det() const noexcept {
    const Vector3D& row0 = row(0);
    const Vector3D& row1 = row(1);
    const Vector3D& row2 = row(2);

    return row0.x() * (row1.y() * row2.w() - row1.w() * row2.y())
         - row0.y() * (row1.x() * row2.w() - row1.w() * row2.x())
         + row0.w() * (row1.x() * row2.y() - row1.y() * row2.x());
}

bool Matrix3x3::hasInverse() const noexcept {
    return det() != 0.0f;
}

Matrix3x3 Matrix3x3::transpose() const noexcept {
    return Matrix3x3(
        Vector3D(column(0)[0], column(0)[1], column(0)[2]),
        Vector3D(column(1)[0], column(1)[1], column(1)[2]),
        Vector3D(column(2)[0], column(2)[1], column(2)[2])
    );
}

Matrix3x3 Matrix3x3::inverse() const noexcept {

    if (det() != 0.0f) {
        float fInverseDetM = 1.0f / det();

        const Vector3D& row0 = row(0);
        const Vector3D& row1 = row(1);
        const Vector3D& row2 = row(2);

        float fInverseElement00 = (row1.y() * row2.w() - row1.w() * row2.y()) * fInverseDetM;
        float fInverseElement01 = -(row0.y() * row2.w() - row0.w() * row2.y()) * fInverseDetM;
        float fInverseElement02 = (row0.y() * row1.w() - row0.w() * row1.y()) * fInverseDetM;

        float fInverseElement10 = -(row1.x() * row2.w() - row1.w() * row2.x()) * fInverseDetM;
        float fInverseElement11 = (row0.x() * row2.w() - row0.w() * row2.x()) * fInverseDetM;
        float fInverseElement12 = -(row0.x() * row1.w() - row0.w() * row1.x()) * fInverseDetM;

        float fInverseElement20 = (row1.x() * row2.y() - row1.y() * row2.x()) * fInverseDetM;
        float fInverseElement21 = -(row0.x() * row2.y() - row0.y() * row2.x()) * fInverseDetM;
        float fInverseElement22 = (row0.x() * row1.y() - row0.y() * row1.x()) * fInverseDetM;

        return Matrix3x3(Vector3D(fInverseElement00, fInverseElement01, fInverseElement02),
                         Vector3D(fInverseElement10, fInverseElement11, fInverseElement12),
                         Vector3D(fInverseElement20, fInverseElement21, fInverseElement22));
    }
    else {
        std::cout << "Determinate of M is Zero";
    }
}
```