Transformación lineal

Kevin del Castillo Ramírez

1. Transformación lineal

```cpp
#ifndef MAC_LINEAR_TRANSFORMATION_HPP
#define MAC_LINEAR_TRANSFORMATION_HPP

#include <matrix.hpp>
#include <except.hpp>

namespace mac {

    using namespace std;

    class LinearTransformation {
    private:
        size_t dimension_in;
        size_t dimension_out;
        Matrix transformation_matrix;

        bool configured = false;

    public:
        LinearTransformation(size_t dimension_in, size_t dimension_out, Matrix matrix = Matrix())
            dimension_in(dimension_in),
            dimension_out(dimension_out)
        {
            set_transformation_matrix(matrix);
        }

        void set_transformation_matrix(Matrix matrix) {
            if (matrix.n_rows != dimension_out || matrix.n_cols != dimension_in)
                throw IncorrectDimensionParameter();

            transformation_matrix = matrix;
            configured = true;
        }

        void set_dimensions_parameters(size_t dimension_in, size_t dimension_out) {
            this->dimension_in = dimension_in;
            this->dimension_out = dimension_out;
            configured = false;
        }

        CVector operator()(CVector const& input) {
            if (!configured)
                throw NotConfigured();

            if (input.n_rows != dimension_in)
                throw IncorrectDimensionParameter();

            return transformation_matrix * input;
        }

        RVector operator()(RVector const& input) {
            if (!configured)
                throw NotConfigured();

            if (input.n_cols != dimension_in)
                throw IncorrectDimensionParameter();

            return transformation_matrix * CVector(input);
        }
    };
}

#endif
```

2. Código de prueba

```
#include <iostream>
#include <linear_transformation.hpp>

using namespace std;
using namespace mac;

int main() {
    LinearTransformation t(3, 2);

    Matrix transformation_matrix({
        { 1, 0, 1 },
        { 1, 1, 2 }
    });

    t.set_transformation_matrix(transformation_matrix);

    cout << t(CVector({ 2, 3, 1 })) << '\n';
}
```

```cpp
#include <cmath>
#include <iostream>
#include <linear_transformation.hpp>

using namespace std;
using namespace mac;

int main() {
    Matrix polygon({
        { 2, 2 },
        { 4, 2 },
        { 4, 4 },
        { 2, 4 }
    });

    cout << "Initial polygon (each row is a vertice)\n";
    cout << polygon;

    LinearTransformation rotate(2, 2, Matrix({
        { cos(M_PI/2), -sin(M_PI/2) },
        { sin(M_PI/2),  cos(M_PI/2) }
    }));

    for (auto i = 0; i < polygon.n_rows; ++i)
        polygon.row(i) = rotate(RVector(polygon.row(i)));

    cout << "Rotated polygon (each row is a vertice)\n";
    cout << polygon;
}
```