Transformacje geometryczne

Kolejne zadanie stanowi przerywnik w tworzeniu elementów interfejsu użytkownika. Przygotowuje jednak infrastrukturę do wygodnej implementacji animacji.
Tematem zadania piątego będzie implementacja klasy reprezentującej punkt w przestrzeni dwuwymiarowej (znany z poprzednich ćwiczeń `FPoint`) oraz szablon klasy reprezentującej macierz transformacji geometrycznych (`Matrix`).
Dodatkowym elementem, który będzie wymagał nieco uwagi, są testy klasy i szablonu. Testy nie będą oceniane, jednak będę je przeglądać (i komentować). Żeby nie było jednak różnic w definicjach klasy i szablonu, powtórzymy je poniżej.

Klasa FPoint i funkcje „obliczeniowe”

```
class FPoint
{
public:
    float x,y;
    FPoint(float cx=0.0f, float cy=0.0f) : x(cx), y(cy) {}
    friend std::ostream& operator <<(std::ostream& os, const FPoint& p);
    friend std::istream& operator >>(std::istream& is, FPoint& p);
    operator Graph_lib::Point() const;
};

FPoint min(const FPoint& lf, const FPoint& rt);
FPoint max(const FPoint& lf, const FPoint& rt);
FPoint operator+(const FPoint& lpoint, float val);
FPoint operator-(const FPoint& lpoint, float val);
FPoint operator+(const FPoint& lpoint, const FPoint& rpoint);
FPoint operator*(const FPoint& lpoint, const FPoint& rpoint);
float distance(const FPoint& p1, const FPoint& p2);
```

Szablon klasy `Matrix`

```
template <typename T> class Matrix
{
    static const int size = 3;
    std::array<std::array<T, size>, size> cf;
    void setIdentity();
    void copyMx(const Matrix& rhm);
public:
    Matrix() { setIdentity(); }
    Matrix(const Matrix& rhm) { copyMx(rhm); }
    Matrix& operator = (const Matrix& rhm) { copyMx(rhm); return *this; }

    static Matrix scaleMx(T scaleX, T scaleY);
    static Matrix translateMx(T offsetX, T offsetY);
    static Matrix rotateMx(T angle);

    friend std::ostream& operator <<(std::ostream& os, const Matrix& mx);
    friend Matrix operator * (const Matrix& lhm, const Matrix& rhm);
    Matrix& operator *= (const Matrix& rhm);
    FPoint transform(const FPoint& pt) const;

    const T& operator()(unsigned int rowid, unsigned int colid) const;
    T& operator()(unsigned int rowid, unsigned int colid);
};
```

Oddawanie

Rozwiązanie zadania składa się z 4 plików:

`fpoint.h` – zawiera definicję i implementację `FPoint`

`fpoint.cpp` – zawiera implementacje metod operujących na `FPoint` zdefiniowanych poza klasą

`matrix.h` – implementacja szablonu klasy `Matrix`

`zad_5.cpp` – testy klasy `FPoint` oraz szablonu `Matrix`.

Rozwiązanie (spakowane w archiwum zip) proszę złożyć w Moodle do:

7 maja 2023 23:59