Compound operators

```
/*

        Compound operators:
            . examples:
                . operator+=
                . operator-=
            . others:
                . operator*=
                . shift operators

*/
```

Compound operators

```
export class Point
{
  friend std::ostream &operator<<(std::ostream &os, const Point &p);
  friend Point operator+(const Point &left, const Point &right);
  friend Point operator-(const Point &left, const Point &right);
  friend Point &operator+=(Point &left, const Point &right);
  friend Point &operator-=(Point &left, const Point &right);

public:
  Point() = default;
  Point(double x, double y) : m_x(x), m_y(y) {}
  ~Point() = default;

private:
  double length() const;// Function to calculate distance from the point(0,0)

private:
  double m_x{};
  double m_y{};
};
```

Compound operators

```
//Implementations
inline Point &operator+=(Point &left, const Point &right)
{
  left.m_x += right.m_x;
  left.m_y += right.m_y;
  return left;
}

inline Point &operator-=(Point &left, const Point &right)
{
  left.m_x -= right.m_x;
  left.m_y -= right.m_y;
  return left;
}


inline Point operator+(const Point &left, const Point &right)
{
  Point p(left);
  return p += right;
}

inline Point operator-(const Point &left, const Point &right)
{
  Point p(left);
  return p -= right;
}
```

Compound operators

```cpp
Point p1(10, 10);
Point p2(20, 20);
Point p3(5, 5);

auto p1_plus_p2 = p1 + p2;
auto p2_minus_p3 = p2 - p3;

std::cout << "p1 + p2: " << p1_plus_p2 << "\n";
std::cout << "p2 - p3: " << p2_minus_p3 << "\n";

std::cout << "------" << "\n";

p1 += p2;
p2 -= p3;

std::cout << "point1: " << p1 << "\n";
std::cout << "point2: " << p2 << "\n";
```