Prefix and postfix increment operators

```
/*

        Topics:
            . Unary prefix increment operator (member/non-member)
            . Unary postfix increment operator (member/non-member)
            . There also are
                    . decrement versions of these operators
                    . you can set them up as an exercise.

*/
```

Prefix operators

```cpp
//The goal
fix_ops_1::Point p1(10, 10);
std::cout << "p1: " << p1 << "\n";
++p1;
p1.operator++(); //the operator is a member
std::cout << "p1: " << p1 << "\n";
```

```cpp
//The goal
  fix_ops_2::Point p1(10, 10);
  std::cout << "p1: " << p1 << "\n";
  ++p1;
  operator++(p1); // the operator is a non-member
  std::cout << "p1: " << p1 << "\n";
```

Prefix and postfix increment operators

```
//Unary prefix operator+ as member
export class Point
{
friend std::ostream &operator<<(std::ostream &os, const Point &p);

public:
Point(double x, double y) : m_x(x), m_y(y) {}

//prefix operator done as a member
void operator++()
{
    ++m_x;
    ++m_y;
}

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

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

Prefix and postfix increment operators

```
//Unary prefix operator+ as non-member
export class Point
{
friend std::ostream &operator<<(std::ostream &os, const Point &p);
friend void operator++(Point &operand);

public:
Point(double x, double y) : m_x(x), m_y(y) {}

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

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

inline std::ostream &operator<<(std::ostream &os, const Point &p)
{
os << "Point [ x: " << p.m_x << ", y: " << p.m_y << "]";
return os;
}

//Unary prefix operator+ done as a non-member
inline void operator++(Point& operand){
        ++(operand.m_x);
        ++(operand.m_y);
}
```

Postfix operators

```
fix_ops_3::Point p1(10, 10);
std::cout << "p1: " << p1++ << "\n";
std::cout << "p1: " << p1 << "\n";
```

Prefix and postfix increment operators

```
//Unary postfix operator+ as member
export class Point
{
friend std::ostream &operator<<(std::ostream &os, const Point &p);

public:
Point(double x, double y) : m_x(x), m_y(y) {}

//Prefix operator done as a member
void operator++()
{
    ++m_x;
    ++m_y;
}

//Postfix done as a member. Done in terms of prefix operator
Point operator++(int){
    Point local_point(*this);
    ++(*this);
    return local_point;
}

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

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

Prefix and postfix increment operators

```
//Unary postfix operator+ as a non-member
export class Point
{
friend std::ostream &operator<<(std::ostream &os, const Point &p);
friend void operator++(Point &operand);
friend Point operator++(Point &operand, int);

public:
Point(double x, double y) : m_x(x), m_y(y) {}

private:

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

//Prefix operator+ done as a non-member
inline void operator++(Point& operand){
    ++(operand.m_x);
    ++(operand.m_y);
}

//Postfix operator++ done as non-member. In terms of prefix operator
Point operator++(Point &operand, int)
{
    Point local_point(operand);
    ++operand;
    return local_point;
}
```