2023 09 06

```
// operatr ++ ve --  overload edilmesi
/*
    increment
    decrement

    ++x     prefix increment
    x++     postfix increment

    --x     prefix decrement
    x--     postfix decrement

    int y = 10;
    auto b = y++;  b = 10

    y = 10
    auto z = ++y;  z == 11

*/
```

```
class Counter {
    public:
        Counter& operator++(); // prefix
        Counter operator++(int); // postfix

        Counter& operator--(); // prefix
        Counter operator--(int); // postfix
};

int main()
{
    int a[] = { 1, 2, 3, 4, 5 };
    int *p = a;

    std::cout << *p++ << "\n";  // 1
    std::cout << *p << "\n";  // 2
    std::cout << *++p << "\n";  // 3
}
```

operator + ve – overload

```
// operator + ve - overload
class Nint
{
    public:
        Nint operator+()const
        {
            return *this;
        }
        Nint operator-()const
        {
            return Nint(-mx);
        }
};
```

operator[]

```
class String {
    public:
        String(const char *p) : mp{ new char[std::strlen(p)  1]}
        {
            std::strcpy(mp, p);
        };
        std::size_t length() const
        {
            return std::strlen(mp);
        }
        friend std::ostream& operator<<(std::ostream& os, const String& s)
        {
            return os << " " << s.mp << "'";
        }
        char& operator[](std::size_t idx)
        {
        }
    private:
        char *mp;
};

int main()
{
    String str{"mustafa demirhan"};
    std::cout << str[1] << "\n";

    str[0] = '!';

    for (size_t i{}; i < str.length(); ++i)
    {
        std::cout << str[i] << ' ';
    }
}
```