Statikus tagok

- statikus függvények

```
class Counter
{
        static unsigned c;
public:
        Counter(){c++;}
        ~Counter(){c--;}

        static unsigned getCounter() { return c;}
};

//cpp
unsigned Counter :: c = 0;

//Main
…
Counter :: getCounter(); //itt is scope operátort kell használni! nem kötődik az
                                objektum példányaihoz; NEM kapják meg a this pointert!
                                nem fér hozzá csak statikus tagokhoz!
```

- strukturált felületek függvényei(qsort)
- ISR – interrupt service routine

Operátorok túlterhelése

a = c++; → visszatérési értéke (= jel után) / mellékhatásként c megnövelődik eggyel

a = b + c * d; → precedencia
a = c* d * e; → asszociativitás!

a = b = 3; → itt már jobbról balra van az asszociativitási irány

```
int fv(int &c)
{
        int a = c;
        c = c + 1;
        return a;
}
```

c++ és a fv(c) C++-ban ugyanaz!

```
int prefix_plspls(int &c)
{
        c = c+1;
        return c;
}
```

A + B ⇔ operator+(a,b) // ez működik C++-ban

Complex z;

z + a ⇔ operator+(z, a);

Complex operator+(Complex z, double a) ➔ itt írjuk meg az operátor függvényt
{
...
}

class Complex
{
…
Complex operator+(double a)
{
…
}
…
};

Complex osztály

(1) Complex :: Re(z)
Complex :: Im(z)
Complex :: Conjugate(z)

(2) 2 + j*3 // j = i ☺

(3) z = z1 + z2;
z = z1- z2;
…

(3) 3 + z
3* z
3/z
…

(4) cout << z1 << z2;
cin >> z1;

[kiadott lap]

~~Standard iostream

cout << 3.2 << 4.3;
cout << 3.2; ⇔ operator<<(cout, 3.2);

Operator<<(operator<<(cout, 3.2), 4.3); //visszatérése cout (objektuma iostreamnek)

Dinamikus osztályok

operátor = **másolókonstruktor**

← dinamikus osztálynak írni kell →
alapértelmezett (sekély)

értékadás
a = b;

már inicializált

inicializálni kell
int a = b;

dinamikus osztálynak kell:
- konstruktor
- destruktor
- copy konstruktor

általános szabályok:
- csak beépített típusokterhelhetőek túl
- . , :: , ?: és a sizeof() nem terhelhetőek felül

pl.:
a.) Mátrix:

```
Matrix m (2,1);
m(1, 1) = 3;
cout << m(1,1);

Matrix
{
        double &operator()(unsigned x, unsigned y) {…};
        double operator()(unsigned x, unsigned y)const {}
};
```

b.)

Counter:

```
int operator++(){…} //++c felel meg
int operator++(int) {…} //c++ felel meg
```