\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

saat 21:00

->iki string arasında +, += çalışıyo ama -, -= çalışmıyo

->string "[]" (stringde overloadlanmış subscript operatörü) boundary check yapmıyor(stringin uzunluğunu aştın mı uyarı vermiyor):

string x = "alaz"; x[15] = "a" gibi bir işlem yapmamaya dikkat etmem lazım

x.at(15); yazınca out of range exception atıyor(programi crashliyor ve hatayi yazıyor)

-> yani string.at() operatörü boundary check yapıyor.

->c++ yeni operatör oluşturmaya izin vermiyor, fakat var olan operatörleri overloadlamaya izin veriyor

->operatör overloading otomatik değil, kendim yapmam lazım

->classlarda overloaded operatör function kullanmak için bu overloaded operatörleri definelamam lazım, şu 3üne dikkat:

**operator=** ----> "=" i overloadlerken "="in sağındaki objenin datalarını "="in solundaki objenin içine kopyalarız. fakat eğer classta pointer varsa, bu kopyalama tehlikeli olabilir(shallow copy). yani kopyalarken sadece sağ objedeki pointerin işaret ettiği adres sol objenin pointerine kopyalanırsa, bir objenin pointeri üzerinde yaptığımız işlem diğer objede de uygulanmış olur. kısaca dinamik bellek yönetiminde problem oluşturur.

**operator(&)** ----> objeyi point eden bir pointer returnler // address operator

**operator(,)** ----> virgül operatörü önce virgülün solundaki ifadeyi değerlendirir, sonrasında sağı değerlendirir ve sağdaki ifadenin sonucunu geri döndürür:

*int x = (5,10); // output: x = 10* a

-> overloadlanamayan operatörler:

“.” “.\*”(pointer to member) “::” “ ?:”

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*