AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

Szoftver laboratórium 2. 3. ellenőrző dolgozat. 2012.03.26. Kurz/**Terem**: L1/	**15 perc**
Név: **Neptun:**	Összpont:

1.feladat 4.5+1.5 pont

Valósítson meg C++ nyelven egy olyan osztályt (***KomplexTar***), ami tetszőleges számú *Komplex* típusú objektumot képes tárolni! Elvárás az osztállyal szemben, hogy másolható legyen és értékadás bal és jobb oldalán is szerepelhessen. Legyen az osztálynak egy ***keres()*** tagfüggvénye, ami a tárolóból kikeres egy, a paramétereként kapott valós értékkel egyező abszolút értékű komplex számot. Ha talál, akkor az első megtaláltat adja vissza, ha nincs ilyen, úgy egy komplex nullát (*Komplex(0,0)*) adjon. Tételezze fel, hogy a *Komplex* osztály létezik, és helyesen működik!

Megadtuk a ***KomplexTar*** osztály deklarációját, és a tagfüggvények definícióját, de ez több helyen hiányos. A **pontozott** részekre írva **egészítse ki** az alábbi **kódrészletet** úgy, hogy a fenti elvárásoknak megfelelően működjön, és ne lépjen fel memóriakezelési hiba! Nem kell minden pontozott részre írnia! Tételezze fel, hogy az **értékadás operátor,** amely külön állományban van megvalósítva, **helyesen működik,** ezért azt **nem kell** megírnia!

A feladatlap hátoldalán **adja meg**, hogy **megoldása** minimálisan **milyen elvárásokat támaszt** a *Komplex* osztállyal szemben! (pl,: van olyan konstruktora, ami két valós számot vesz át; van … operátora; stb.) (1.5p)

```
class KomplexTar {
    Komplex* pKompl;  // dinamikus adatterület kezdőcíme, ahol tárolunk
    int db;           // tárolt számok száma (pontosan ennyi komplex van)
public:
    // Konstruktor: n darab valós számpárt (2*n darab számot) vesz át, amiből
    // n darab komplex értéket készít és letárolja
    KomplexTar(const double v[], int n = 0) :db(n) {
        pKompl = new Komplex[db];
        for (int i = 0; i < 2*db; i += 2)
            pKompl[i/2] = Komplex(v[i], v[i+1]);
    }
    KomplexTar(const KomplexTar&);
    KomplexTar& operator=(const KomplexTar&);
    Komplex keres(double) const;
    ........~KomplexTar() { delete [] pKompl; }
};
KomplexTar::KomplexTar(const KomplexTar& k) :db(k.db) {
        pKompl = new Komplex[db];
        for (int i = 0; i < db; i++)
            pKompl[i] = k.pKompl[i];
}// Keres: az első megtalált abs abszolút értékű komplex számot adja vissza.
Komplex KomplexTar::keres(double abs) const {
    for (int i = 0; i < db; i++)
        if (pKompl[i].abs() == abs)
            return pKompl[i];
    return Komplex(0,0);
}// Példa a használatra:
int main() {
    double dv[] = { 1, 2, 3, 4, 5, 6, 7, 8 };
    KomplexTar k1(dv, 4);  // négy darab valós számpár (8 darab szám)
    KomplexTar k2 = k1;
    k1 = k2 = k2;
    cout << k2.keres(1.0); // az 1 abs. értékű számot keressük
}
```

Komplex osztállyal szemben támasztott elvárások:

- legyen létrehozható két valós számmal (valós és képzetes rész)
- legyen default konstruktora
- legyen másoló konstruktora
- legyen értékadás (operator=) operátora
- legyen egy abs() tagfüggvénye, ami visszaadja a szám abszolút értékét
- kiírható legyen egy ostream objektumra