AAAAAAAAAAAAAAAAAAAAAAAAAAAA

Programozás alapjai II. 4. ellenőrző dolgozat. 2012.04.19. Kurz/**Terem**: G1/	**20 perc**
Név: **Neptun:**	Összpont:

1. feladat 4 pont

Egy **térinformatikai rendszerben** különböző objektumokat akarunk egy tárolóban (*Tarolo*) tárolni. Tudjuk, hogy legfeljebb 300 objektumunk lehet. Minden objektumnak van egy azonosítója (*String*), amit tárolni kell, de a további adatok teljesen eltérőek lehetnek. Jelenleg csak utcát (*Utca*) és benzinkutat (*Kut*) akarunk tárolni, de később bővíteni akarjuk a rendszert. Az utcát a neve azonosítja, de tároljuk a hosszát is. A benzinkút esetében a neve mellett tároljuk az is, hogy hétvégén nyitva van-e. A rendszerben megvalósítandó műveleteket egy kódrészlettel mutatjuk be:

```
Tarolo proba;                          // Ez lesz az tárolónk
proba.uj(new Utca("Irinyi", 2000)); // Utca hozzáadása (utcanév és hossz)
proba.uj(new Kut("OMV", true));      // Benzinkút (neve és nyitva van)
proba.kiir();              // tárolóból minden adatot kiírunk (név, hossz, stb.)
proba.clear();                         // Összes tárolt adatot töröljük
```

Feltételezheti, hogy a *Tarolo* osztályból példányosított objektumot nem akarjuk paraméterként átadni és értékadás jobb, ill. bal oldalán sem szerepel. **Tervezzen** meg olyan osztályhierarchiát, ami alkalmas az objektumok tárolására és könnyen bővíthető. Ügyeljen arra, hogy jól határozza meg az objektumok felelősségét! **Deklarálja** a *Tarolo*, *Utca*, és *Kut* osztályokat! **Csak** a *Tarolo* osztály konstruktorát, valamint az *uj()* és *kiir()* metódusát **valósítsa** meg!

```
class Azon {
   String az;
public:
   Azon(String);
   virtual void kiir();
   virtual ~Azon();
};

class Utca :public Azon {
   double hossz;
public:
   Utca(String, double);
   void kiir();
};

class Kut :public Azon {
   bool nyitva;
public:
   Kut(String, bool);
   void kiir();
};
```

```
class Tarolo {
   Azon* azon[300];
   int db;
public:
   Tarolo() :db(0) {}
   void uj(Azon* p) {
       azon[db++] = p;
   }
   void kiir() {
       for (int i = 0; i < db; i++)
              azon[i]->kiir();
   }
   void clear();
   ~Tarolo();
};
```

---

2. feladat 2 pont

Tételezze fel, hogy a *Tarolo* osztálynak nincs *kiir()* metódusa, de van iterátora, amivel egymás után sorban elérhetők a tárolt elemek. Mutassa be, egy kódrészleten, hogy ezzel az iterátorral hogyan tudná a *kiir()* metódus funkcióját kiváltani!

```
for (Tarolo::iterator i = proba.begin(); i != proba.end(); ++i)
      (*i)->kiir();
```