Adatszerk. vizsga, 2. tétel 2014

Objektumorientált programozás - osztályok és objektumok

Az objektumok a modellezendő valós világ egy-egy önálló egységét jelölik, az objektumokat a **számunkra** lényeges tulajdonságok alapján megkülönböztetjük. Üzeneteket küldhetünk nekik, amikre valamilyen módon reagálhatnak.

Az objektumokat osztályokba soroljuk, úgy, hogy a hasonló tulajdonságokkal rendelkező objektumok egy osztályba kerülnek, míg az eltérő tulajdonságokkal rendelkező objektumok külön osztályokba kerülnek. Az osztályok az objektumok **mintáinak** tekinthetők, hiszen hordozzák az egyes objektumokra jellemző tulajdonságokat.

Objektum: belső állapota van, ebben információt tárol (adattagokkal valósítjuk meg). Az objektum metódusainak hatására változhat meg belső állapota. Minden objektum egyértelműen azonosítható.

Osztály: az objektum mintájának tekinthető, ez alapján hozzuk létre az egyes példányokat.

Objektum létrehozása, inicializálása

Az objektum életciklusa: megszületik, él, meghal. Az objektum inicializálását a konstruktor végzi: adatok kezdőértékadása, objektum működéséhez szükséges tevékenységek végrehajtása, típusinvariáns beállítása.

C++-ban a következő szabályok érvényesek a konstruktorra: neve megegyezik az osztály nevével, ha már megadtunk egy konstruktort, akkor a default konstruktor nem definiálódik, a default konstruktor meghívja az attribútumok konstruktorát, de a beépített típusokat nem inicializálja, a konstruktornak nem lehet visszatérési értéke.

Destruktorok C++-ban: explicite lehet hívni a delete operátorral, vagy implicit hívódik a blokkból való kilépéskor. Fontos: ha a konstruktorban dinamikusan lefoglalunk egy memóriaterületet (new), akkor a destruktorban ezt fel kell szabadítani (delete).

Példányváltozó, példánymetódus, osztályváltozó, osztálymetódus

Példányváltozó: példányonként helyet foglaló változó, az osztály objektumainak állapotleírója.
Példánymetódus: példányokon dolgozó metódus, az osztály objektumainak küldhető üzenetek.
Osztályváltozó: osztályonként helyet foglaló változó, az osztálynak, mint objektumnak az állapotleírója.
Osztálymetódus: osztályokon dolgozó metódus, az osztálynak, mint példánynak küldhető üzenet.

Öröklődés, polimorfizmus, dinamikus összekapcsolás (példákkal)

Öröklődés: Létrehozhatunk ős-osztályokat és utód-osztályokat. Az öröklés során az utódok öröklik őseik metódusait és változóit: az ősosztály minden metódusa és adattagja a gyerekosztálynak is metódusa és adattagja lesz. A leszármazott bevezethet új adattagokat és metódusokat, ezek egyszerűen hozzáadódnak az örökölt adattagokhoz és metódusokhoz. Az utód ezen felül felüldefiniálhatja az örökölt metódusokat, a hierarchiában ezek felüldefiniálják az örökölt metódust.

Polimorfizmus: más néven többalakúság azt jelenti, hogy egy bizonyos típusként deklarált változó a program futása során más típusú változó értékét is felveheti. A statikus típus az a típus, a változó eredeti típusa (így deklaráltuk), míg a dinamikus típus a változó éppen aktuális típusa (ez változhat a program futása során. C++-ban pl.: legyen az employee egy őstípus, míg a manager ennek egy leszármazott típusa. Ekkor a következő kód értelmes és lefut:

```
Employee* empp=new Employee("Istvan", "Nagy", 5);
Manager* mp=new Manager("Laszlo", "Kovacs", 2, 3);
empp=mp;
```

Dinamikus összekapcsolás: Run-time fogalom. Az a jelenség, hogy a változó éppen aktuális értékének megfelelő metódus implementáció hajtódik végre. Az előző példa alapján, ha az `empp.print()` függvényt az `empp=mp` értékadás előtt hívjuk meg, akkor az őstípus metódusa fog lezajlani, ha azonban az értékadás után futtatjuk le a `empp.print()` kódrészletet, akkor már a Manager osztály `print()` függvénye hívódik meg. C++-ban a felüldefiniálható függvényeknél az őstípusban a virtual kulcsszót kell használni.

Absztrakt osztály, a többszörös öröklődés problémái, lehetséges megoldásai

Absztrakt osztály: olyan osztály, amely a tervezés eszköze, nem hozható létre példánya, mivel a leszármazott teszi konkréttá. A metódusai között van olyan, amelynek csak specifikációja van, törzse nincs. Absztrakt osztály az, amelyben van legalább egy pure virtual függvény: `virtual [függvény szignatúra]=0`

A többszörös öröklődés problémái, megoldásai: A többszörös öröklődés azt jelenti, hogy egy típusnak akár több típus is lehet az őse, ekkor mindkét típustól örökli a tulajdonságokat. De a többszörös öröklődésnek lehetnek problémái:

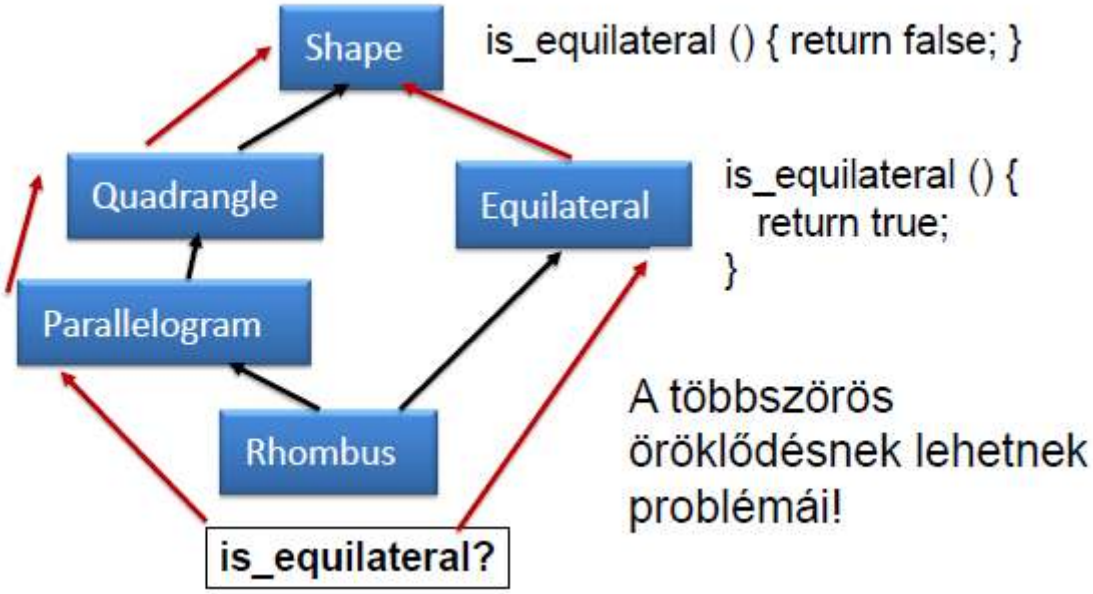

Figure 1: A Shape ős nem egyenlő oldalú, míg az Equilateral egyenlő oldalú. A Rhombus egyenlőoldalú? Nem egyértelmű!

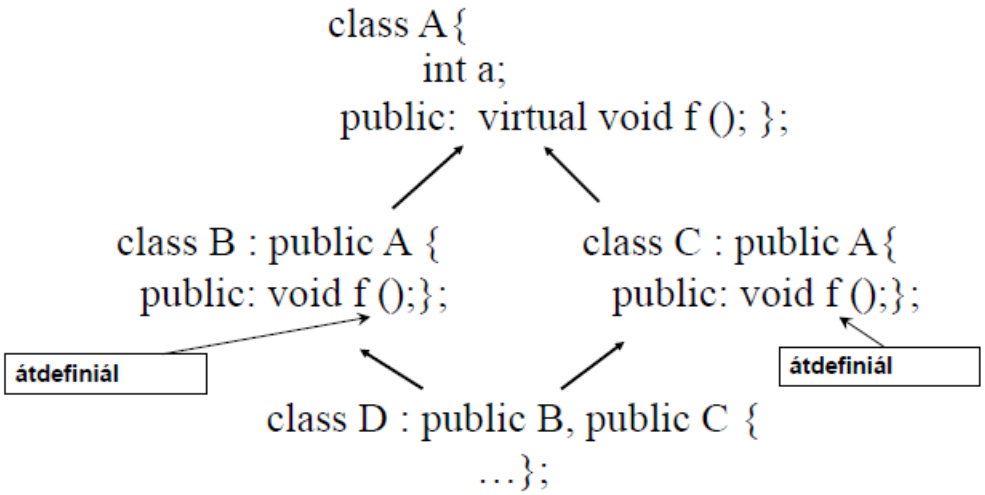

Figure 2: Ha egy D-beli 'f'-re hivatkozunk, akkor az melyiket jelentse? Az 'a' attribútum hány példányban jelenjen meg D-ben?

A második ábrán látható esetben legtöbbször a kódot nem lehet lefordítani, a fordító a kétértelműségre (ambiguous) hivatkozva leáll. Megoldási variációk: az ősosztály mondja meg, hogy mit szeretne tenni ilyen

esetben, vagy a származtatott osztály mondja meg, hogy melyiket szeretné használni. Pl.:

```
class D: public B, public C
{
public:
using C::f;
};
```

Arra a problémára, hogy 'a' attribútum hányszor jelenjen meg D-ben, a következő a megoldás: Az A osztály B és C virtuális bázisosztálya kell legyen, ekkor D a-t(minden adattagot) csak egyszer örökli.