A C++ másoló-mozgató szemantikáját tesztelő feladat

A feladat megoldásához négy metódust kellett implementálni egy saját osztályban, melyek a másoló konstruktor, a mozgató konstruktor, a másoló értékadás, és a mozgató értékadás. Ahhoz, hogy meg tudjuk oldani a feladatot, négy dolgot kell tudnunk:

1. a konstruktor egy objektum **létrejöttekor fut le**
2. az értékadás a **már létrejött** objektumokon operál
3. mi az az **rvalue** – olyan objektumok, amik (legtöbbször) nem nevesítettek, és a **kifejezés végére éréskor megsemmisülnek**
4. mi az az **lvalue** – olyan objektumok, amik (legtöbbször) nevesítettek, és a **kifejezés végére éréskor nem semmisülnek meg**

Azt, hogy a négy metódus közül mikor melyik fut le, az fogja eldönteni, hogy mikor jön létre az objektum, és hogy ideiglenes (azaz rvalue)-e.

A fontosabb a két szempont közül az l/r value értéke, ugyanis ez dönti el, hogy az eredeti objektumot le kell másolnunk, vagy mozgathatjuk a benne lévő adatokat. Ha egy érték rvalue, akkor nincsen rá másik hivatkozás azon kívül, amit mi birtoklunk paraméternévként, így a benne tárolt adatokhoz soha senki nem férhet hozzá, tehát nem baj, ha azok megszűnnek, áthelyeződnek, vagy átíródnak. Ebben az esetben sokkal hatékonyabb a (viszonylag) lassú másolásnál, ha az adatokat átvesszük a paramétertől mielőtt az megsemmisül, azonban ha az érték lvalue, értékéhez más is hozzáférhet, így azt meg kell őriznünk akkor is, ha a másolás drágább művelet.

Hogy az implementációnk sikeres volt-e, néhány egyszerű kódsorral ellenőrizhetjük:

* operátorok (pl. +, -, \*) és függvényhívások rvalue-t adnak vissza, így ezekben az esetekben a mozgató értékadásnak kell meghívódnia
* ha egy változóhoz egy másikat név szerint hozzárendelünk, a másoló értékadásnak kell megtörténnie
* Ha konstruktort hívunk, akkor hasonlóképpen annak paramétereként megadhatunk akár nevesített változót másoláshoz, akár operátorok vagy függvények eredményét mozgatáshoz

Ahhoz, hogy a mozgató konstruktort a mozgató értékadásra alapozzuk, meg kell őriznünk a (immár nevesített, így lvalue) kapott paraméter rvalue mivoltát, melyet C++-ban az std::move(1) függvénnyel tehetünk meg.

A konkrét implementáció egy Labda osztályt tartalmaz, melynek az std::string szín és az int méret tulajdonságát másoljuk le és helyezzük át a négy metódusban. Mind a négy kiírja a standard kimenetre, mikor lépett be a vezérlés a megadott metódusba, valamint van egy Labda értéket visszaadó függvény, ami ezt szintén megteszi. A main(0) metódus olyan kódot tartalmaz, melyek Labda változókkal az l/r value és a konstruálás/értékadás különböző kombinációit hívják meg.