7. Gyakorlat

Öröklés, kivételkezelés

1. Definiáljunk egy `Teglalap` osztályt, és származtassuk ebből a `Negyzet` osztályt, melyek téglalapokat ill. négyzeteket reprezentálnak. A `Teglalap` osztály adattagjai reprezentálják a magasságot ill. szélességet, és rendelkezzen a következő tagfüggvényekkel:

   - konstruktor két paraméterrel,
   - függvények a magasság és szélesség lekérdezésére és megváltoztatására,
   - függvények a terület és kerület kiszámítására.

   A `Negyzet` osztály rendelkezzen egy egyparaméterű konstruktorral, amely a `Teglalap` osztály konstruktorán alapszik. Ezen felül legyenek ugyanazok a tagfüggvényei, mint a `Teglalap` osztálynak, de a magasságot ill. a szélességet megváltoztató függvények változtassák meg mindkettőt. Mely tagfüggvényeket érdemes virtuális függvényként megadni (és miért)?

2. Mit ír ki a képernyőre következő kód és miért? Oldjuk meg a feladatot a kód futtatása nélkül, majd futtassuk is a kódot.

```
#include <iostream>
using namespace std;

class base {
public:
    void fun_1() { cout << "base-1\n"; }
    virtual void fun_2() { cout << "base-2\n"; }
    virtual void fun_3() { cout << "base-3\n"; }
    virtual void fun_4() { cout << "base-4\n"; }
};

class derived : public base {
public:
    void fun_1() { cout << "derived-1\n"; }
    void fun_2() { cout << "derived-2\n"; }
    void fun_4(int x) { cout << "derived-4\n"; }
};

int main() {
    base* p;
    derived obj1;
    p = &obj1;

    p->fun_1();
    p->fun_2();
    p->fun_3();
    p->fun_4();

    return 0;
}
```

3. Definiáljunk egy `Alakzat` osztályt, amely egy síkidomot reprezentál. Legyenek adattagjai, melyek a síkidom pozícióját határozzák meg. Származtassunk belőle különböző osztályokat, pl. kör, téglalap, paralelogramma, sokszög stb. osztályokat. Legyenek az `Alakzat` osztálynak tisztán virtuális `terulet` és `kerület` tagfüggvényei, és valósítsuk meg ezeket a származtatott osztályokban. Deklaráljunk olyan tömböt/tárolót, amelyben különböző típusú síkidomokat tárolhatunk.

4. Készítsünk egy matematikai kivételek kezelésére szolgáló osztályt, amely az `std::exception` leszármazottja. Származtassunk ebből további, speciális hibák esetén (pl. 0-val való osztás, szinguláris objektummal végzett tilos művelet, nem megfelelő dimenziókból származó hiba) dobandó kivételeket. Írjuk felül a `what()` függvényt, mely szövegesen leírja az adott hibát.

   Írjuk át a korábban írt `Rational` osztályt, amely racionális számokat tárol. Dobjunk kivételt, ha a konstruktorban a nevezőt 0-ra állítják, ha 0-val akarnak osztani, ill. ha a 0-nak akarják a reciprokát venni. Használjuk a fent készített megfelelő kivételosztályt.

5. Írjunk $m \times n$-es mátrixokat reprezentáló template osztályt (ahol a template paraméter az elemek típusa). Írjunk konstruktort, ami paraméterként a mátrix méreteit kapja meg. Az elemhozzáférést megvalósító operátor ill. tagfüggvény implementációjában helytelen indexek esetén dobjunk kivételt. A mátrixhoz lehessen hozzáadni, belőle kivonni más $m \times n$-es mátrixokat, valamint más mátrixszal is össze lehessen szorozni, amennyiben a dimenziók nem megfelelőek a dobjunk kivételt. Írjunk függvényt, amely négyzetes mátrix esetén kiszámolja a mátrix inverzét, 0 determináns esetén dobjunk kivételt (elég 2×2-es mátrix esetén megvalósítani). A kivételkezeléshez használjuk az előző feladatban készített kivételosztályokat.