A rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 4. ellenőrző dolgozat. 2016.04.25. Kurz/**Terem**: G1/	Elért pontszám:
Név: **Neptun:**	

1. feladat 2 p

Számítógép tartozékokat (*Printer*, *Monitor*, *Lapolvaso*, stb.) szeretnénk heterogén kollekcióban tárolni. Ehhez elkészítettünk egy absztrakt alaposztályt ***Tartozek*** néven. A *print()* tagfüggvény szolgál a különféle adatok a szabványos kimenetre történő kiírására. Kollégája átadta papíron Önnek az alaposztály deklarációját, de ráömlött a kávé. Ennyi maradt meg belőle. Reprodukálja az osztályt!

```
struct Tartozek {

    virtual void print() const = 0;

    virtual ~Tartozek() {}
};
```

2. feladat 5 p

Tételezze fel, hogy rendelkezésére áll a gyakorlatokon elkészített iterátoros generikus tömb (*Array<T, int siz = 6>*). A tömb indexelhető és van *size()* metódusa, ami az elemek aktuális számát adja. Hozzon létre a generikus tömb felhasználásával egy *Raktar* osztályt, ami képes maximum 1000 darab, az előző feladatban reprodukált alaposztályból származó tartozékot heterogén gyűjteményként tárolni! Legyen a *Raktar* osztálynak *add()*, és *list()* metódusa, amivel új tartozékot tehetünk a raktárba, illetve ki tudjuk listázni minden raktárban levő tartozék fontosabb jellemzőjét. A *Raktar* osztály adatait közvetlenül ne lehessen elérni! A felsoroltakon és az implicit módon is keletkező tagfüggvényeken kívül más publikus függvénye ne legyen! Amennyiben egy *Raktar* típusú objektum értékparaméterként átadásra kerülne, akkor dobjon *std::invalid_argument* hibát! Az alábbi kódrészlet a használatra mutat példát:

```
Raktar enKisRaktaram;
enKisRaktaram.add(new Monitor);
enKisRaktaram.add(new Printer);
enKisRaktaram.list();
```

Egy lehetséges megoldás tartalmazott objektummal (delegáció). Kihasználtuk, hogy az indexelés nyújtja a tömböt, ahogyan a gyakorlaton látott példában. Természetesen a tényleges méretet saját adminisztrációval is lehetett volna kezelni:

```
class Raktar {
    Array<Tartozek*, 1000> tar;
public:
    Raktar() {}   // a másoló miatt kell
    Raktar(const Raktar&) { throw std::invalid_argument("Nono"); }
    void add(Tartozek *t) { tar[tar.size()] = t; }
    void list() {
       for (size_t i = 0; i < tar.size(); i++)
           tar[i]->print();
    }
    ~Raktar() {
       for (size_t i = 0; i < tar.size(); i++)
           delete tar[i];
    }
};
```

3. feladat 3 pont

Rövid kódrészlettel mutassa be az előző feladatban elkészített *Raktar* osztály használatát úgy, hogy az dobjon *std::invalid_argument* hibát! Kapja el a kivételt és írja ki a szabványos hibakimenetre a kivétel által szállított szöveget!

Egy lehetséges megoldás:

```
try {
    Raktar r1;
    Raktar r2 = r1;
} catch (std::exception& e) {
    std::cerr << e.what() << std::endl;
}
```

A rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 4. ellenőrző dolgozat. 2016.04.25. Kurz/**Terem**: G1/	Elért pontszám:
Név: **Neptun:**	

Raktár örökléssel (megjegyzést ld. a 2. feladatnál)

```
class Raktar :private Array<Tartozek*, 1000> {
public:
    Raktar() {}
    Raktar(const Raktar&) { throw std::invalid_argument("Nono"); }
    void add(Tartozek *t) { (*this)[size()] = t; }
    void list() {
       for (size_t i = 0; i < size(); i++)
           (*this)[i]->print();
    }
    ~Raktar() {
       for (size_t i = 0; i < size(); i++)
           delete (*this)[i];
    }
};
```

BBBBBBBBBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 4. ellenőrző dolgozat. 2016.04.25. Kurz/**Terem**: G1/	Elért pontszám:
Név: **Neptun:**	

1. feladat 2 p

Különféle üdülőhelyek jellemzőit (*Camping, Diakszallo, Hotel*, stb.) szeretnénk heterogén kollekcióban tárolni. Ehhez elkészítettünk egy absztrakt alaposztályt ***Udulo*** néven. A *write()* tagfüggvény szolgál a különféle adatok szabványos kimenetre történő kiírására. Kollégája átadta papíron Önnek az alaposztály deklarációját, de ráömlött a kóla. Ennyi maradt meg belőle. Reprodukálja az osztályt!

```
class Udulo {
public:
      virtual void write() const = 0;
      virtual ~Udulo() {}
};
```

2. feladat 5 p

Tételezze fel, hogy rendelkezésére áll egy a gyakorlatokon elkészített generikus tömbhöz (*Array<T, int siz = 100>*) hasonló osztály, melynek van *at()* metódusa az elemek eléréséhez és *size()* metódusa, ami az elemek aktuális számát adja. Hozzon létre a generikus tömb felhasználásával egy *Lista* osztályt, ami képes maximum 200 darab, az előző feladatban reprodukált alaposztályból származó üdülőhelyet heterogén gyűjteményként tárolni! Legyen a *Lista* osztálynak *ujHely()*, és *megmutat()* tagfüggvénye, amivel új üdülőhelyet adhatunk a listához, illetve meg tudjuk mutatni a listában levő üdülőhelyek jellemzőit. A *Lista* osztály adatait közvetlenül ne lehessen elérni! A felsoroltakon és az implicit módon is keletkező tagfüggvényeken kívül más publikus függvénye ne legyen! Amennyiben a listánk értékadó operátorát használnák, akkor dobjon *std::invalid_argument* hibát! Az alábbi kódrészlet a használatra mutat példát:

```
Lista udulohelyeim;
udulohelyeim.ujHely(new Hotel);
udulohelyeim.ujHely(new Camping);
udulohelyeim.megmutat();
```

Egy lehetséges megoldás tartalmazott objektummal (delegáció). Kihasználtuk, hogy az at() nyújtja a tömböt, ahogyan a gyakorlaton látott példában. Természetesen a tényleges méretet saját adminisztrációval is lehetett volna kezelni:

```
class Lista {
    Array<Udulo*, 200> tar;
public:
    Lista& operator=(const Lista&) { throw std::invalid_argument("Huh"); }
    void ujHely(Udulo *u) { tar.at(tar.size()) = u; }
    void megmutat() {
       for (size_t i = 0; i < tar.size(); i++)
           tar.at(i)->write();
    }
    ~Lista() {
       for (size_t i = 0; i < tar.size(); i++)
           delete tar.at(i);
    }
};
```

3. feladat 3 pont

Rövid kódrészlettel mutassa be az előző feladatban elkészített *Lista* osztály használatát úgy, hogy az *std::invalid_argument* hibát dobjon! Kapja el a kivételt és írja ki a szabványos hibakimenetre a kivétel által szállított szöveget!

Egy lehetséges megoldás:

```
try {
    Lista l1, l2;
    l2 = l1;
} catch (std::exception& e) {
    std::cerr << e.what() << std::endl;
}
```