A rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 4. ellenőrző dolgozat. 2016.04.25. Kurz/**Terem**: G2/	Elért pontszám:
Név: **Neptun:**	

1. feladat 2 p

Villanyszerelési szerszámokat (*Multimeter, Fazisceruza, Krimpelo,* stb.) szeretnénk heterogén kollekcióban tárolni. Ehhez elkészítettünk egy absztrakt alaposztályt ***Szerszam*** néven. A *jellemez()* tagfüggvény szolgál a szerszám jellemzőinek szabványos kimenetre történő kiírására. Kollégája átadta papíron Önnek az alaposztály megvalósítását, de ráfolyt a tintapatron. Ennyi maradt meg belőle. Reprodukálja az osztályt!

```
struct Szerszam {

    virtual void jellemez() const throw(const char*) = 0;

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

---

2. feladat 5 p

Rendelkezésére áll a gyakorlatokon elkészített generikus tömb (*Array<T, size_t maxsiz = 98>*). A tömb indexelhető és van *size()* metódusa, ami az elemek aktuális számát adja. Hozzon létre a generikus tömb felhasználásával egy *Keszlet* osztálysablont, ami képes sablonparaméterként megadott mennyiségű, az előző feladatban reprodukált alaposztályból származó szerszámot heterogén gyűjteményként tárolni! Legyen a *Keszlet* sablonnak *betesz()*, és *felsorol()* metódusa, amivel új szerszámokat tehetünk a szerszámkészletbe, illetve ki tudjuk listázni minden készletbeli szerszám jellemzőjét. A *Keszlet* adatait közvetlenül ne lehessen elérni! Kapja el a *jellemez()* tagfüggvény által dobott kivételt és dobjon helyette *std::out_of_range* hibát! Az alábbi kódrészlet a használatra mutat példát:

```
Keszlet<16> ladam;        // 16 db szerszám fér a ládába
ladam.betesz(new Fazisceruza);
ladam.betesz(new Multimeter);
ladam.felsorol();
```

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

```
template<size_t db>
class Keszlet {
    Array<Szerszam*, db> tar;
public:
    void betesz(Szerszam *s) { tar[tar.size()] = s; }
    void felsorol() {
       try {
         for (size_t i = 0; i < tar.size(); i++)
             tar[i]->jellemez();
       } catch (const char* s) { throw std::out_of_range(s); }
    }
    ~Keszlet() {
       for (size_t i = 0; i < tar.size(); i++)
           delete tar[i];
    }
};
```

---

3. feladat 3 pont

Hozzon létre egy *KivetelTeszt* osztályt, ami eltárolható a fenti heterogén kollekcióban. Az osztály a jellemzők kiírása helyett *const char** kivételt dobjon! Tételezze fel, hogy a *ladam* objektum tárol ilyen objektumot! Mutassa meg a keletkező kivétel elkapását!

Egy lehetséges megoldás:

```
struct KivetelTeszt : Szerszam {
    void jellemez() const throw(const char*) { throw "Hiba"; }
};

try {
    ladam.felsorol();
} 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**: G2/	Elért pontszám:
Név: **Neptun:**	

Készlet örökléssel (megjegyzést ld. a 2. feladatnál)

```
template<size_t db>
class Keszlet : private Array<Szerszam*, db> {
public:
    void betesz(Szerszam *s) { (*this)[Array<Szerszam*, db>::size()] = s; }
    void felsorol() {
       try {
         for (size_t i = 0; i < Array<Szerszam*, db>::size(); i++)
             (*this)[i]->jellemez();
       } catch (const char* s) { throw std::out_of_range(s); }
    }
    ~Keszlet() {
       for (size_t i = 0; i < Array<Szerszam*, db>::size(); i++)
           delete (*this)[i];
    }
};
```

BBBBBBBBBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 4. ellenőrző dolgozat. 2016.04.25. Kurz/**Terem**: G2/	Elért pontszám:
Név: **Neptun:**	

1. feladat 2 p

Különféle megjelenítő eszközök (*LCD, EInk, Projector*, stb.) jellemzőit szeretnénk heterogén kollekcióban tárolni. Ehhez elkészítettünk egy absztrakt alaposztályt ***DisplayDevice*** néven. A *show()* tagfüggvény szolgál a megjelenítő jellemzőinek szabványos kimenetre történő kiírására. Kollégája átadta papíron Önnek az alaposztály megvalósítását, de a papír elázott az esőben. Ennyi maradt meg belőle. Reprodukálja az osztályt!

```
class DisplayDevice {
public:
      virtual void show()const throw(bool) = 0;
      virtual ~DisplayService() {}
};
```

2. feladat 5 p

Rendelkezésére áll egy a gyakorlatokon elkészített generikus tömbhöz (*Array<T, size_t maxsiz = 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 *Storage* osztálysablont, ami képes sablonparaméterként megadott mennyiségű, az előző feladatban reprodukált alaposztályból származó megjelenítőt heterogén gyűjteményként tárolni! Legyen a *Storage* sablonnak *addItem()* és *showAll()* tagfüggvénye, amivel új megjelenítőket adhatunk a tárolóhoz, illetve meg tudjuk mutatni a tárolóban levő megjelenítők jellemzőit. A *Storage* adatait közvetlenül ne lehessen elérni! Kapja el a *show()* tagfüggvény által dobott kivételt és dobjon helyette *std::out_of_range* hibát! Az alábbi kódrészlet a használatra mutat példát:

```
Storage<20> mydisplays;       // 20 megjelenítőt befogadó tároló
mydisplays.addItem(new Projector);
mydisplays.addItem(new LCD);
mydisplays.showAll();
```

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

```
template<size_t db>
class Storage {
    Array<DisplayDevice*, db> tar;
public:
    void addItem(DisplayDevice *d) { tar.at(tar.size()) = d; }
    void showAll() {
       try {
         for (size_t i = 0; i < tar.size(); i++)
             tar.at(i)->show();
       } catch (bool) { throw std::out_of_range("Hiba"); }
    }
    ~Storage() {
       for (size_t i = 0; i < tar.size(); i++)
           delete tar.at(i);
    }
};
```

3. feladat 3 pont

Hozzon létre egy *TestExcetion* osztályt, ami eltárolható a fenti heterogén kollekcióban. Az osztály a jellemzők kiírása helyett *bool* kivételt dobjon! Tételezze fel, hogy a *mydisplays* objektum tárol ilyen objektumot! Mutassa meg a keletkező kivétel elkapását!

Egy lehetséges megoldás:

```
struct TestException : DisplayDevice {
    void show() const throw(bool) { throw false; }
};
try {
    mydisplays.showAll();
} catch(std::exception& e) {
    std::cerr << e.what() << std::endl;
}
```