A rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 4. ellenőrző dolgozat. 2016.04.25. Kurz/**Terem**: G3/	Elért pontszám:
Név: **Neptun:**	

1. feladat 2 p

Különféle weboldalak (*Kereso, Hiroldal, Blog*, stb.) jellemzőit szeretnénk heterogén kollekcióban tárolni. Ehhez elkészítettünk egy absztrakt alaposztályt ***Weboldal*** néven. Az osztály függvényhívás operátora szolgál a weboldal adatainak szabványos kimenetre történő kiírására. Kollégája átadta papíron Önnek az alaposztály megvalósítását, de véletlenül bedarálta a hűtőventilátor. Ennyi maradt meg belőle. Reprodukálja az osztályt!

```
struct Weboldal {

        virtual void operator()() const = 0;

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

---

2. feladat 5 p

Rendelkezésére áll a gyakorlatokon elkészített iterátoros generikus tömb (*Array<T, size_t maxsiz = 6>*). A tömb indexelhető és van *size()* metódusa, ami az elemek aktuális számát adja. Hozzon létre a generikus tömb felhasználásával egy *HeteroTar* osztálysablont, ami képes maximum 163 darab sablonparaméterként megadott alaposztályból származó objektumot heterogén gyűjteményként tárolni! Legyen a *HeteroTar* sablonnak *uj()*, és *kiir()* metódusa, amivel új objektumot tehetünk a gyűjteménybe, illetve ki tudjuk íratni a gyűjteményben levő objektumok fontosabb jellemzőit. A jellemzők kiírásához a sablonparaméterként megadott alaposztály függvényhívás operátorát kell meghívni. A *HeteroTar* adatait közvetlenül ne lehessen elérni!

```
HeteroTar<Weboldal> myBookmarksFolder;
myBookmarksFolder.uj(new Blog);
myBookmarksFolder.uj(new Hiroldal);
myBookmarksFolder.kiir();
```

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:

```
template <class T>
class HeteroTar {
    Array<T*, 163> tar;
public:
    void uj(T* www) { tar[tar.size()] = www; }
    void kiir() {
       for (size_t i = 0; i < tar.size(); i++)
           (*tar[i])();
    }
    ~HeteroTar() {
       for (size_t i = 0; i < tar.size(); i++)
          delete tar[i];
    }
};
```

---

3. feladat 3 pont

Tételezze fel, hogy az előző feladatban elkészített sablont kiegészítettük iterátorral, melynek segítségével bejárható a tároló. Írjon egy kódrészletet, melyben a *myBookmarksFolder* objektumból iterátor segítségével kilistázza weboldalak jellemzőit!

Egy lehetséges megoldás:

```
typedef HeteroTar<Weboldal>::iterator HTiter;
for (HTiter iter = myBookmarksFolder.begin(); iter != myBookmarksFolder.end(); ++iter)
    (*(*iter))();        // vagy  (*iter)->operator()();
```

A rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 4. ellenőrző dolgozat. 2016.04.25. Kurz/**Terem**: G3/	Elért pontszám:
Név: **Neptun:**	

HeteroTar örökléssel (megjegyzést ld. a 2. feladatnál)

```
template <class T>
class HeteroTar :public Array<T*, 163> {
public:
    void uj(T* www) { (*this)[Array<T*, 163>::size()] = www; }
    void kilistaz() {
       for (size_t i = 0; i < Array<T*, 163>::size(); i++)
           (*(*this)[i])();
    }
    ~HeteroTar() {
       for (size_t i = 0; i < Array<T*, 163>::size(); i++)
           delete (*this)[i];
    }
};
```

HStore (B csoport) örökléssel (megjegyzést ld. a 2. feladatnál)

```
template <class T>
class HStore : public Array<T*, 15> {
public:
    void add(T *tt) { Array<T*, 15>::at(Array<T*, 15>::size()) = tt; }
    void list() {
       for (size_t i = 0; i < Array<T*, 15>::size(); i++)
           (*Array<T*, 15>::at(i))();
    }
    ~HStore() {
       for (size_t i = 0; i < Array<T*, 15>::size(); i++)
           delete Array<T*, 15>::at(i);
    }
};
```

BBBBBBBBBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:18 **perc**

Programozás alapja 2. 4. ellenőrző dolgozat. 2016.04.25. Kurz/**Terem**: G3/	Elért pontszám:
Név: **Neptun:**	

1. feladat 2 p

Egyetemi tantárgyak (*AlapkepzesesTargy*, *SzakiranyosTargy*, *ValaszthatoTargy*, stb.) szeretnénk heterogén kollekcióban tárolni. Ehhez elkészítettünk egy absztrakt alaposztályt ***Tantargy*** néven. Az osztály függvényhívás operátora szolgál a tárgy jellemzőinek egy paraméterként kapott *std::ostream* típusú objektumra történő kiírására. Kollégája átadta papíron Önnek az alaposztály megvalósítását, de kiégette a forrasztópáka. Ennyi maradt meg belőle. Reprodukálja az osztályt!

```
class Tantargy {
public:

        virtual void operator(std::ostream&)() const = 0;

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

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 *HStore* osztályt, ami képes maximum 15 darab sablonparaméterként megadott alaposztályból származó objektumot heterogén gyűjteményként tárolni! Legyen a *HStore* osztálynak *add()* és *list()* tagfüggvénye, amivel új objektumot tehetünk a gyűjteménybe, illetve ki tudjuk íratni egy std::ostream típusú objektumra a gyűjteményben levő objektumok fontosabb jellemzőit. A jellemzők kiírásához a sablonparaméterként megadott alaposztály függvényhívás operátorát kell meghívni. A *HStore* adatait közvetlenül ne lehessen elérni! Az alábbi kódrészlet a használatra mutat példát:

```
HStore<Tantargy> masodik;
masodik.add(new AlapkepzesesTargy);
masodik.add(new ValaszthatoTargy);
masodik.list(std::cout);
```

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:

```
template <class T>
class HStore {
    Array<T*, 15> tar;
public:
    void add(T *tt) { tar.at(tar.size()) = tt; }
    void list(std::ostream& os) {
       for (size_t i = 0; i < tar.size(); i++)
           (*tar.at(i))(os);
    }
    ~HStore() {
       for (size_t i = 0; i < tar.size(); i++)
           delete tar.at(i);
    }
};
```

3. feladat 3 pont

Tételezze fel, hogy az előző feladatban elkészített sablont kiegészítettük iterátorral, melynek segítségével bejárható a tároló. Írjon egy kódrészletet, melyben a *masodik* objektumból iterator segítségével kiírja a szabványos hibakimenetre a tantárgyak jellemzőit!

Egy lehetséges megoldás:

```
typedef HStore<Tantargy>::iterator HSiter;
for (HSiter iter = masodik.begin(); iter != masodik.end(); ++iter)
    (*(*iter))(std::cerr);  /// vagy (*iter)->operator()(std::cerr);
```