AAAAAAAAAAAAAAAAAAAAAAAAAAAA

Programozás alapjai II. 5. ellenőrző dolgozat. 2012.05.03. Kurz/**Terem**: G2/	**20 perc**
Név: **Neptun:**	Összpont:

1. feladat 3 pont

Tételezze fel, hogy rendelkezésére áll egy sorozattároló (*Vector*)! A tároló a szokásos műveletekkel rendelkezik (*push_back, pop_back, front, back, size, ...*), csak default konstruktora van. Sajnos nem jelez semmilyen hibát, ha egy üres tároló utolsó elemét akarjuk elérni (back). Ezen osztály felhasználásával készítsen egy olyan generikus osztályt (*MyVector)*, ami *out_of_range()* hibát dob ilyen esetben, egyébként a *Vector* osztállyal teljesen azonos módon működik! Működjön helyesen az alábbi programrészlet:

```
MyVector<double> vec;                  // double elemeket tartalmazó tároló
try {
   vec.push_back(11.0);                // 11-et teszünk a tárolóba
   cout << vec.back() << endl;         // kiírjuk a legutolsó elemet
   vec.pop_back();                     // eldobja a legutolsó elemet
   vec.back();                         // hibát dob
} catch (exception& e) {
   cerr << "Kivetel:" << e.what() << endl;
}
```

Örökléssel egyszerűbb, de tartalmazott objektummal is megoldható volt. A size utáni …-tal azt akartuk sugallni, hogy még sok egyéb metódus lehet. Beágyazott objektummal való megvalósításnál ezeket mind delegálni kellett volna.

```
template<typename T>
class MyVector :public Vector<T> {
public:
      T& back() {
        if (size() == 0) throw std::out_of_range("MyVector: back");
            return Vector<T>::back();
      }
};
```

Teljesen precíz a megoldás akkor lenne, ha a

```
      const T& back() const;
```

tagfüggvényt is megvalósítjuk, de ezt nem vártuk el.

---

2. feladat 3 pont

Írjon függvénysablont (***fetolt***), ami az STL *fill* sablonjához hasonlóan a függvény 3. paramétereként megadott értékre állítja egy tároló iterátorokkal kijelölt elemeit. A függvény első két paramétere két iterátor (OutputIterator), ami kijelöli a jobbról nyílt intervallum kezdetét és végét. A függvény *void* visszatérési értékű. Amennyiben helyesen oldja meg a feladatot, akkor az alábbi kódrészlet a következőt írja ki: **10, 2, 2, 2, 2, 2,**

```
    int v[] = { 10, 2, 3, 17, 18, 6 };
    feltolt(v+1, v+6, 2);              // ezt a sablont kell megvalósítania
    copy(v, v+6, ostream_iterator<int>(cout, ", "));// kiírjuk az elemeket
```

```
template<class Iter, class T>
void feltolt(Iter first, Iter last, const T& val) {
    while (first != last)
        *first++ = val;
}
```

---

3. feladat +1 pont

Mire használatos a signal/slot mechanizmus?

A objektumok metódusainak összekapcsolásár való. A származtatással megvalósított callback funkciók kiváltására. A signal/slot mechanizmus megoldás típus biztos,ugyanakkor lazább csatolást biztosít.