BBBBBBBBBBBBBBBBBBBBBBBBBBB

Programozás alapjai II. 5. ellenőrző dolgozat. 2012.05.03. Kurz/**Terem**: G3/	**20 perc**
Név: **Neptun:**	Összpont:

1. feladat 3 pont

Tételezze fel, hogy rendelkezésére áll egy sorozattároló (*Vektor*)! A tároló a szokásos műveletekkel rendelkezik (*push_back, pop_back, front, back, size, ...*), csak default konstruktora van. Szeretnénk, ha lenne egy olyan művelete is (*swap_ends*), ami lehetővé teszi, hogy a sorozat első elemét megcseréljük az utolsó elemével. A *Vektor* osztály felhasználásával készítsen egy olyan generikus osztályt (*Vektorom)*, ami pontosan úgy működik, mint a *Vektor*, és van *swap_ends()* tagfüggvénye! Ez utóbbi dobjon *out_of_range()* hibát, ha a tároló üres, vagy csak 1 eleme van! Működjön helyesen az alábbi programrészlet:

```
Vektorom<double> vec;                  // double elemeket tartalmazó tároló
try {
   vec.push_back(11.0);                // 11-et teszünk a tárolóba
   vec.push_back(33.0);                // 33-at teszünk a tárolóba
   vec.push_back(21.0);                // 21-et teszünk a tárolóba
   vec.swap_ends();                    // megcseréli a 11-et és a 21-et
   vec.pop_back();                     // eldobja a legutolsó elemet
   vec.pop_back();                     // eldobja a legutolsó elemet
   vec.pop_back();                     // eldobja a legutolsó elemet
   vec.swap_ends();                    // 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 Vektorom :public Vektor<T> {
public:
   void swap_ends() {
      if (size() <= 1) throw std::out_of_range("MyVector: back");
      T tmp = back();
      back() = front();
      front() = tmp;
   }
};
```

---

2. feladat 3 pont

Írjon függvénysablont (***myreplace_if***), ami az STL *replace_if* sablonjához hasonlóan a függvény 4. paramétereként megadott értékre állítja egy tároló azon elemeit, melyek a 3. paraméterben átadott predikátumnak eleget tesznek! 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: **1, 2, 3, 8, 8, 8,**

```
  int v[] = { 1, 2, 3, 17, 18, 6 };
  myreplace_if(v, v+6, bind2nd(greater<int>(), 5), 8);//ezt kell megvalósítani
  copy(v, v+6, ostream_iterator<int>(cout, ", "));// kiírjuk az elemeket
```

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

---

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.