AAAAAAAAAAAAAAAAAAA rendelkezésre álló idő: **15 perc**

Szoftver laboratórium 2.	5. Ellenőrző dolgozat.	2014.04.28.	Kurz/**Terem**: L1/
Név:	**Neptun:**		Összpont:

1. feladat 2.5 pont

Írjon generikus függvényt (***kivalaszt***), ami kiválaszt egy adott tulajdonságú elemet egy sorozattárolóból! Az adott tulajdonságot kétoperandusú függvénnyel (predikátummal) határozza meg! A sablon és a függvény paramétereit, visszatérési értékét Önnek kell meghatároznia úgy, hogy a 2. feladatban azt hatékonyan fel tudja használni!

```
template <class Iter, class Pred>
Iter kivalaszt(Iter elso, Iter utolso, Pred cmp) {
    Iter tmp = elso;
    while(++elso != utolso)
        if (cmp(*elso, *tmp)) tmp = elso;
    return tmp;
}
```

---

2. feladat 3.5 pont

Tételezze fel, hogy rendelkezésére áll egy generikus tároló (pl. a korábbi laborgyakorlatokon elkészített *Array<T>*)! Mutassa be ennek felhasználásával az elkészített sablon használatát a következő kódrészletek megírásával:

1. Hozzon létre egy *egész* és egy *std::string* elemeket tartalmazó tárolót!
2. Tételezze fel, hogy létezik egy 100 elemű vektor (*int vektor[100]*)! Töltse fel az egész elemek tartalmazó tárolót a vektor elmeivel iterátorok használatával!
3. Az 1. feladatban elkészített generikus algoritmus felhasználásával írja ki a szabványos kimenetre az egész tároló legnagyobb elemét! Készítsen ehhez megfelelő predikátumot!
4. Szintén a generikus algoritmus felhasználásával írja ki a szabványos kimenetre az *std::string* elemeket tároló tárolóban levő leghosszabb stringet! Készítsen ehhez megfelelő predikátumot! Feltételezheti, hogy létezik a *size_t td::string::length() const;* függvény, ami az adott string hosszát adja..

// függvénysablonnal + specializációval

```
template <class T>
bool haNagyobb(const T& a, const T& b) { return a > b; }

template <>
bool haNagyobb(const std::string& a, const std::string& b) {
    return a.length() > b.length(); }

Array<int> intTar(vektor, vektor+100);
Array<std::string> strTar;

std::cout <<*kivalaszt(intTar.begin(), intTar.end(), haNagyobb<int>) << std::endl;
std::cout <<*kivalaszt(strTar.begin(),strTar.end(),haNagyobb<std::string>)<< std::endl;
```

// természetesen más módon is előállíthatóak a predikátum függvények, pl. függvénysablon nélkül:

```
bool intNagyobb(int a, int& b) { return a > b; }

bool strNagyobb(const std::string& a, const std::string& b) {
    return a.length() > b.length(); }
```

// a tömb feltöltése pedig assign tagfüggvénnyel is elvégezhető:

```
Array<int> intTar;
intTar.assign(vektor, vektor+100);


std::cout <<*kivalaszt(intTar.begin(), intTar.end(), intNagyobb) << std::endl;
std::cout <<*kivalaszt(strTar.begin(), strTar.end(), strNagyobb) << std::endl;
```