AAAAAAAAAAAAAAAAAAA rendelkezésre álló idő: 15 **perc**

Programozás alapja 2. 4. ellenőrző dolgozat. 2015.04.20. Kurz/**Terem**: G1/	Elért pontszám:
Név: **Neptun:**	

1. Írjon C++ függvénysablont (*keres*), ami egy iterátorokkal megadott adatsorozatban megkeresi az első olyan elemet, amire az adott predikátum igaz értéket ad! 4p
   A függvény első két paramétere két iterátor, ami kijelöli az adatsorozat kezdetét és végét (jobbról nyílt intervallum). A függvény 3. paramétere pedig a predikátum, ami egy egyparaméteres függvény vagy függvényobjektum. Amennyiben nincs a feltételnek megfelelő elem, akkor az adatsorozat végét jelző értékkel (iterátor) térjen vissza a függvény! Ha jól oldja meg a feladatot, akkor az alábbi kódrészlet lefutása után az *eredmény* a 3-as indexű elemre (-16) fog mutatni.

```
bool negativ(int a) { return a < 0; }
int sorozat[] = { 1, 4, 9, -16, 25};      // a sorozat
int *eredmeny = keres(sorozat, sorozat+5, negativ);
```

2. A korábbi gyakorlaton elkészített iterátoros generikus tömb (Array) felhasználásával hozzon létre egy *double* értékeket, és egy *long* értékeket tartalmazó tömböt! A 1. feladatban elkészített sablon és az Array sablon iterátoros interfészének felhasználásával keresse meg az első 0 (nulla) elemet a *long* értékeket tartalmazó tömbben! 3p
3. Készítsen olyan függvényobjektum sablont, ami a *keres* sablonnal felhasználható egy olyan elem megkeresésére, amely megegyezik a generikus adat paraméter nélkül hívható konstruktora által létrehozott adattal! Tételezze fel, hogy van a generikus adatnak == operátora és paraméter nélkül hívható konstruktora! 2p
4. Az 1. és a 3. feladatban elkészített sablonok felhasználásával keresse meg a 2. feladatban készített *long* elemeket tartalmazó tömbben az első 0 (nulla) elemet! 1p

Egy lehetséges megoldás:

```
template <class Iter, class P>
Iter keres(Iter first, Iter last, P pred) {
    while (first != last){
        if (pred(*first)) return first;
        ++first;
    }
    return first;
}

Array<double> da1;
Array<long> la1;
bool nulla(long a) { return a == 0; }
keres(la1.begin(), la1.end(), nulla);

template <class T>
struct Azonos {
    bool operator()(T a) { return a == T(); }
}

keres(la1.begin(), la1.end(), Azonos<long>());
```

BBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:15 **perc**

Programozás alapja 2. 4. ellenőrző dolgozat. 2015.04.20. Kurz/**Terem**: G1/	Elért pontszám:
Név: **Neptun:**	

1. Írjon C++ függvénysablont (*holvan*), ami egy iterátorokkal megadott adatsorozatban megkeresi az első olyan elemet, amire az adott predikátum igaz értéket ad! 4p
   A függvény első két paramétere két iterátor, ami kijelöli az adatsorozat kezdetét és végét (jobbról nyílt intervallum). A függvény 3. paramétere pedig a predikátum, ami egy egyparaméteres függvény vagy függvényobjektum. Amennyiben nincs a feltételnek megfelelő elem, akkor az adatsorozat végét jelző értékkel (iterátor) térjen vissza a függvény! Ha jól oldja meg a feladatot, akkor az alábbi kódrészlet lefutása után az *eredmény* a 4-es indexű elemre (`'2'`) fog mutatni.

```
bool szamjegy(int ch) { return ch >= '0' && ch <= '9'; }
char *azon = "nzh_20150420";                          // a sorozat
char *eredmeny = holvan(azon, azon+10, szamjegy);
```

2. A korábbi gyakorlaton elkészített iterátoros generikus tömb (Array) felhasználásával hozzon létre egy char értékeket, és egy int értékeket tartalmazó tömböt! Az 1. feladatban elkészített sablon és az Array sablon iterátoros interfészének felhasználásával keresse meg az első páros elemet az int értékeket tartalmazó tömbben! 3p
3. Készítsen olyan függvényobjektum sablont, ami a *holvan* sablonnal felhasználható egy olyan elem megkeresésére, ami eltér a generikus adat paraméter nélkül hívható konstruktora által létrehozott adattól! Tételezze fel, hogy van a generikus adatnak != operátora és paraméter nélkül hívható konstruktora is! 2p
4. Az 1. és a 3. feladatban elkészített sablonok felhasználásával keresse meg a 2. feladatban készített *int* elemeket tartalmazó tömbben az első nem nulla elemet! 1p

Egy lehetséges megoldás:

```
template <class Iter, class P>
Iter holvan(Iter first, Iter last, P pred) {
    while (first != last){
        if (pred(*first)) return first;
        ++first;
    }
    return first;
}

Array<char> cha1;
Array<int> ia1;
bool paros(int a) { return (a%2) == 1; }
holvan(ia1.begin(), ia1.end(), paros);

template <class T>
struct Elter {
    bool operator()(T a) { return a != T(); }
}

holvan(ia1.begin(), ia1.end(), Elter<int>());
```