AAAAAAAAAAAAAAAAAAA rendelkezésre álló idő: 20 **perc**

Programozás alapja 2. 3. ellenőrző dolgozat. 2015.04.17. Kurz/**Terem**: G3/	Elért pontszám:
Név: **Neptun:**	

1. **Deklaráljon** C++ nyelven rendezetlen generikus véges halmazt (Set)! Tudjuk, hogy a halmaz számossága nem több, mint 284! A következő műveletekkel rendelkezzen az osztály: 7p
   - új elem hozzáadása a halmazhoz (set)
   - adott elem törlése (*erase*)
   - a halmaz összes elemének törlése (clear)
   - annak vizsgálata, hogy egy adat eleme-e a halmaznak (*element*)
   - annak vizsgálata, hogy a halmaz üres-e (*empty*)

   Az elemtörlés (*erase*) **kivételével valósítson** meg minden műveletet! A tároló adatai közvetlenül ne legyenek elérhetők! Hibakezeléssel nem kell foglalkoznia! Használatára az alábbi kódrészlet mutat példát:

```
using std::cout;
Set<int> ti;              // 284 elem kapacitású üres halmaz létrehozása
Set<char> tc;             // üres halmaz karakterek tárolására
tc.set('A');              // betesszük az 'A' betűt-t
tc.set('A');              // újból, de nem változik semmi, mivel ez halmaz!
tc.set('Z');              // betesszük a 'Z' betűt-t
cout << tc.element('A'); // az 'A' benne van
tc.erase('A');            // kivesszük az 'A'-t
cout << tc.element('A'); // most már nincs benne
cout << tc.empty();       // nem üres, mert a 'Z' még benne van
tc.clear();               // most már üres
```

2. Az előző feladatban megvalósított sablont specializálja úgy, hogy ha a generikus típus double, akkor az element tagfüggvény mindig igaz értékel térjen vissza! 3p

megoldás:

```
template <class T>
class Set {
    static const int M=284;
    T data[M];
    int db;
public:
    Set() :db(0) {}
    void set(const T& d) { if (!element(d)) data[db++] = d; }
    void erase(const T& d); // nem kellett megvalósítani
    void clear() { db = 0; }
    bool element(const T& d) const {
        for (int i = 0; i < db; i++)
            if (data[i] == d)
               return true;
        return false;
    }
    bool empty() const { return db == 0; }
};

template <>
bool Set<double>::element(const double&) const { return true; }
```

BBBBBBBBBBBBBBBBBBBBBB rendelkezésre álló idő:20 **perc**

Programozás alapja 2. 3. ellenőrző dolgozat. 2015.04.17. Kurz/**Terem**: G3/	Elért pontszám:
Név: **Neptun:**	

1. **Deklaráljon** C++ nyelven rendezetlen generikus véges halmazt (Halmaz)! Tudjuk, hogy a halmaz számossága nem több, mint 652! A következő műveletekkel rendelkezzen az osztály: **7p**
   - új elem hozzáadása a halmazhoz (*add*)
   - adott elem törlése (*del*)
   - a halmaz összes elemének törlése (*urit*)
   - annak vizsgálata, hogy egy adat eleme-e a halmaznak (*eleme*)
   - a halmazban levő elemek száma (*size*)

   Az elemtörlés (*del*) **kivételével valósítson** meg minden műveletet! A tároló adatai közvetlenül ne legyenek elérhetők! Hibakezeléssel nem kell foglalkoznia! Használatára az alábbi kódrészlet mutat példát:

```
using std::cout;
Halmaz<int> ti;          // 652 elem kapacitású üres halmaz létrehozása
Halmaz<char> tc;         // üres halmaz karakterek tárolására
tc.add('1');             // betesszük az '1' számjegyet
tc.add('1');             // újból, de nem változik semmi, mivel ez halmaz!
tc.add('0');             // betesszük az '1'-et
cout << tc.eleme('1');   // az '1' benne van
tc.del('1');             // kivesszük az '1'-et
cout << tc.eleme('1');   // most már nincs benne
cout << tc.size();       // 1 elem van benne
tc.urit();               // most már üres
```

2. Az előző feladatban megvalósított sablont specializálja úgy, hogy ha a generikus típus *float*, akkor az *eleme* tagfüggvény mindig hamis értékel térjen vissza! **3p**

megoldás:

```
template <class T>
class Halmaz {
    static const int M=652;
    T data[M];
    int db;
public:
    Halmaz() :db(0) {}
    void add(const T& d) { if (!eleme(d)) data[db++] = d; }
    void del(const T& d); // nem kellett megvalósítani
    void urit() { db = 0; }
    bool eleme(const T& d) const {
        for (int i = 0; i < db; i++)
            if (data[i] == d)
               return true;
        return false;
    }
    int size() const { return db; }
};

template <>
bool Halmaz<float>::eleme(const float&) const { return false; }
```