503.AA Megoldo Arisztid 503.A

Programozás alapjai 2. (inf.) 2. ZH 2017.04.27. gy./l. hiány:	/
ABCD123 **IL305/1**	kZH: nZH:0

Minden beadandó megoldását a feladatlapra, a feladat után írja! Készíthet piszkozatot, de csak a feladatlapra írt megoldásokat értékeljük! Az IMSC feladatot az alapfeladatok 75%-os teljesítése mellett értékeljük.

*Feltételezheti, hogy minden szükséges input adat az előírt formátumban rendelkezésre áll. A feladatok megoldásához csak a letölthető C, C++ és STL összefoglaló használható. Elektronikus eszköz nem használható. A feladatokat **figyelmesen olvassa** el, megoldásukhoz **ne használjon fel STL** tárolót, kivéve, ha a feladat ezt külön engedi/kéri! **Ne írjon felesleges** függvényeket ill. kódot!*

Az első feladatrészben minimum 5 pontot el kell érnie ahhoz, hogy a többi feladatot értékeljük.

1. feladat: Beugró A feladatok megoldásához **használhat STL tárolót és algoritmust!** **Σ 10 pont**

a) Mit ír ki az alábbi program a szabványos kimenetre? Válaszát soronként a vonalazott részre írja! (3p)

```
#include <iostream>
using std::cout;
using std::endl;

inline void e() { cout << endl; }

struct A {
   A()         { cout << 'k'; }
   A(const A&) { cout << 'c'; }
   ~A()        { cout << 'd'; }
   void f(A a) { cout << 'f'; }
   A& operator=(const A&){
                 cout << '=';
                 return *this; }
};

int main() {
   A a;     e();//  _________________
   A b = a; e();//  _________________
   a = b;   e();//  _________________
   a.f(b);  e();//  _______________
}              //  _________________
```

b) Mutassa be az STL sorozattárolók létrehozásának (konstruktor) 4 jellemező használati esetét! (paraméterek száma, típusa)! (2p)

Adott az alábbi kódrészlet:

```
std::set<int> zha1;
long *p21 = new long[100];
...// itt további utasítások vannak
```

c) Írjon olyan kódrészletet, ami kiírja a zha1 objektum adatait soronként a szabványos kimenetre! (1p)!

d) Növelje meg a **p21** pointer által hivatkozott adatterületet a duplájára úgy, hogy a már meglevő adatok ne vesszenek el! A már nem használt memóriaterületet szabadítsa fel! (2p)

e) Jelölje (pl. karikázza be), hogy az állítás igaz (I), vagy hamis (H) a C++ nyelvre! Minden bejelölt válasz 0.5 pont, ha helyes, -0.5p pont, ha hibás! Az esetleges negatív eredmény is összeadódik a többi feladatra kapott ponttal. (2p)

Az iteráror egy általánosított pointer.
A kivételosztályokat kötelező az *std::exception* osztályból származtatni.
Alaposztály konstruktorából hívott virtuális tagfüggvény a leszármazottban fut le.
Függvénysablon példányosítása futási időben történik.

2. Feladat **10 pont**

a) **Készítsen** egy generikus halmaz osztályt (*Set*), adott maximális méretű halmaz tárolására! A méretet sablonparaméterként vegye át! Az osztály az implicit módon keletkező tagfüggvényeken kívül csak a következő **tagfüggvényekkel** rendelkezzen: (5p)

size_t size() const; - a tárolóban levő elemek számát adja
size_t capacity() const; - a tároló maximális méretét (sablon paraméter) adja
int count(const T&) const; - megadja, hogy a paraméterként kapott elem benne van-e tárolóban (0 vagy 1)
void insert(const T&); - beteszi a paraméterként kapott elemet a tárolóba, ha még nincs benne.

Amennyiben már nincs több hely a tárolóban dobjon *std::out_of_range* kivételt! A használatra az alábbi kódrészlet mutat példát:

```
Set<int, 20> fix_halmaz;
fix_halmaz.insert(3);
fix_halmaz.insert(3);
std::cout << fix_halmaz.size();   // kiir: 1
std::cout << fix_halmaz.count(4); // kiir: 0
```

b) Az elkészített sablon felhasználásával, annak módosítása nélkül **készítsen** egy újabb sablont *MySet* néven! Ez legyen kompatibilis a *Set* sablonnal, valamint legyen olyan iterátoros konstruktora is, ami iterátorokkal adott intervallum adataival tölti fel. Ügyeljen arra, hogy üres halmazt is létre lehessen hozni, és ne töltse túl a tárolót! Azt az adatot, ami nem fér bele, hagyja figyelmen kívül! A tárolóban levő elemek számát ne adminisztrálja többszörösen! (3.5p)

c) Rövid kódrészlettel **mutassa be** a *MySet* sablon iterátoros konstruktorának és az *insert* tagfüggvényének használatát *int* adatokkal. A kódrészletben mutassa be az esetlegesen keletkező kivétel elkapását is! (1.5p)

d) IMSC FELADAT:
Egészítse ki a Set sablont iterátorral! Valósítsa meg az összes olyan tagfüggvényt, amit az alábbi kódrészlet felhasznál!

```
for (Set<int, 20>::iterator it = fix_halmaz.begin(); it != fix_halmaz.end();)
    cout << *it++ << endl;
```

---

3. Feladat **10 pont**

a) Valósítson meg egy olyan osztályt (*Student*), ami hallgatók nevének és tanulmányi átlagának tárolására alkalmas! A neveket *std::string* a pontszámokat pedig valós (*double*) típusban tárolja. Mindkét adat legyen megadható a konstruktorban és külön-külön legyen lekérdezhető, de közvetlenül ne legyenek elérhetőek! Elképzelhető, hogy az osztályt később alaposztálynak használjuk heterogén gyűjteményben! (3p)

b) Valósítson meg egy olyan osztályt (*LabGroup*), ami egy csoportnevet (*std::string*) és a csoporthoz tartozó hallgatók adatait tárolja. A hallgatók (*Student*) tárolásához válasszon megfelelő tárolót az STL tárolói közül! A *LabGroup* osztálynak legyen olyan tagfüggvénye (*add*), amivel *Student* adható a csoporthoz és legyen egy olyan is (*list*), amivel a hallgatók adatai **névsorban** kiírathatók a paraméterként kapott *std::ostream* típusú adatfolyamra! Az adatokat tetszőleges sorrendben tárolhatja, amit bármikor át is rendezhet. Az adattagok közvetlenül ne legyenek elérhetőek! Az osztály használatára az alábbi kódrészlet mutat példát, melyből látnia kell, hogy a dinamikus területen létrejövő objektumok felszabadításáról is kell gondoskodnia. (7p)

```
LabGroup l1("L1R4I");      // csoport neve
l1.add(new Student("Fellebeki Atmegel", 4.5));
l1.add(new Student("Bond James Bond", 5));
l1.list(std::cout);
```

4. Feladat **10 pont**

A Honvédelmi Minisztérium HR osztálya a minisztériumi dolgozók nyilvántartására alkalmas szoftvert fejleszt. A nyilvántartásban (*Nyilvantartas*) heterogén kollekciót valósítanak meg. Minden dolgozót a *Dolgozo* osztály (vagy leszármazottja) reprezentálja. Minden dolgozónak van neve (*string*): A katonák (*Katona*) a rendfokozattal (*rang*) rendelkeznek, az orvosoknál (*Orvos*) nyilvántartják a doktori cím megszerzésének idejét (*date*). Vannak olyanok, aki egyszerre katonák és orvosok (*Katonaorvos*). A *KatonaOrvos* kompatibilis mind a *Katona*, mind az *Orvos* osztállyal.

Az attribútumok *getter* metódusokkal elérhetők, de privát láthatóságúak és konstruktorból inicializálhatóak.

A dolgozók és leszármazottaik a *kiir* metódusban kiírják a paraméterül kapott *std::ostream* típusú objektumra az összes adatukat, akár többször is, de *getter* metódust nem használhatnak!

A nyilvántartásból több példány is készülhet. Ha egyet beolvasztunk egy másikba, akkor az első kiürül, a benne levő adatok átkerülnek a másikba. A nyilvántartás ki tudja írni a nyilvántartottak adatait egy paraméterként kapott ostream-re (*listáz*).

Tervezzen objektum-orientált megoldást a fenti leírás alapján a dőlt betűs megnevezések felhasználásával! Az attribútumok kivétel nélkül legyenek privátok és konstruktorban állíthatók. A megoldásban használja ki az STL adta lehetőségeket!

Az alábbi kódrészlet mutatja az egyes metódusok és konstruktorok paraméterezését és használatát.

```
Nyilvantartas ny3per1; // egyik nyilvántartás
Nyilvantartas ny3per2; // másik nyilvántartás
ny3per1.felvesz(new Katona("Kovács Jolán", rang("őrnagy")));
ny3per2.felvesz(new KatonaOrvos("Szentfazéky Eutanázia",
                rang("bőrmester"), date(1992, 06,30)));    // Sz.E.-t lajstromba vesszük
Orvos* ko = new KatonaOrvos("Vastag Béla", rang("huszados"), date(2012, 06,30));
ny3per1.felvesz(ko); // Bélát lajstromba vesszük
ny3per2.beolvaszt(ny3per1); // ny3per1 adatai átkerülnek ny3per2 -be
ny3per2.listaz(std::cout);  // ny3per2 adatait kilistázzuk stdout -ra
```

503.AA Megoldo Arisztid 503.A

Rajzoljon UML osztálydiagramot, amin szerepeljenek a metódusok is (konstruktorok nélkül). UML jelöléssel jelölje a láthatóságot is.

Definiálja az alábbi osztályokat, de a metódusoknak csak a deklarációját adja meg!

```
class Nyilvantartas
```

```
class Dolgozo
```

```
class Katona
```

```
class Orvos
```

```
class KatonaOrvos
```

Implementálja a *Katonaorvos* összes metódusát és konstruktorát.