111. AA Megoldó Alfréd AA 111.

Programozás alapjai 2. (inf.) pótzárthelyi 2008.05.22. gyakorlat: 0/0	Érdemjegy:
ABCDEF () IB028/102.	Hftest: **12000**

F.	Max.	Elért
1	3	
2	4	
3	5	
4	3	
5	5	
Σ	**20**	

Minden beadandó megoldását a feladatlapra, a feladat után írja! *A megoldások során feltételezheti, hogy minden szükséges input adat az előírt formátumban rendelkezésre áll.*
A feladatok megoldásához csak a letölthető C és C++ összefoglaló használható. Számítógép, notebook, menedzser kalkulátor, organiser, rádiótelefon nem használható.

*A feladatokat **figyelmesen olvassa** el, megoldásukhoz **ne használjon fel STL** tárolót, kivéve, a feladat ezt külön engedi! **Ne írjon felesleges** függvényeket ill. kódot, a feladatra koncentráljon!*
Jó munkát! *Értékelés: 0-8:1, 9-11:2, 12-14:3, 15-17:4, 18-20:5*

1. Feladat 3 pont

Mit ír ki a szabványos kimenetre az alábbi program? Válaszához használja a négyzetrácsos területet!

```
#include <iostream>
using namespace std;
class A {
  const char *s;
public:
  A(const char *s = "A") :s(s)   { cout << s << 'k'; }
  virtual ~A()                   { cout << 'd'; }
};
class B {
  A a;
public:
  B(const char *s ="C++") :a(s)  { cout << 'K'; }
  B(const B& b)                  { cout << 'C'; }
  B(const A)                     { cout << 'M'; }
  ~B()                           { cout << 'D'; }
};
int main() {
  A ab4("ABCDEF");   cout << endl;
  B b1("C++4");      cout << endl;
  B b2 = b1;         cout << endl;
  B b3(ab4);         cout << endl;
  delete new A;      cout << endl;
  return(0);
}
```

A	B	C	D	E	F	k								
C	+	+	4	k	K									
A	k	C												
A	k	M	d											
A	k	d												
D	d	D	d	D	d	d								

Nagyon sokan nem vették figyelembe, hogy ha felüldefiniáljuk a másoló konstruktort, és abban explicit nem hívjuk meg a tartalmazott osztály másoló konstruktorát, akkor a tartalmazott alapértelmezett konstruktora hívódik meg. Előadáson többször elmondtam, laboron pedig gyakoroltuk. A nagyZH ugyanezt a kérdést feszegette csak örökléssel!

A 2. feladat szinte teljesen megegyezett a 3. kisZH egyik feladatával! :)

2. Feladat Σ 4 pont

Írjon generikus függvényt, amely kiválaszt egy paraméterként megadott tulajdonságú elemet a paraméterként kapott iterátorok közötti intervallummal megadott tárolóból! A függvény két előrehaladó iterátort kap paraméterként, ami kijelöli a jobbról nyílt intervallum kezdetét és végét (pl: x.begin(), x.end()), és egy tulajdonságot meghatározó függvényobjektumot. Ez utóbbi egy logikai függvény, vagy egy olyan osztály, melynek van függvényhívás operátora. Ez a függvényobjektum hivatott eldönteni két, a tárolóban tárolt típusú elemről, hogy melyik teljesíti a kiválasztási kritériumot. (2 pont). **Írjon generikus** kiválasztó függvényobjektumot az elkészített generikus függvényhez, ami a legkisebb elem kiválasztásra használható! (1 pont)

Írjon programrészletet, amelyben az elkészített generikus függvényt alkalmazza egy 30 valos adatot tartalmazó közönséges tömbre, melyből a legkisebb elemet akarjuk kiválasztani! (1 pont)

Nem kell teljes programot írnia! De minden használt függvényt, változót, objektumot deklarálni ill. definiálnia kell!

1. rész:

```
template <class T, class I, class S>
T select(I from, I to, S cmp) {
  T x = *from++;
  for (; from != to; from++)
      if (cmp(*from, x)) x = *from;
  return x;
}
```

A szöveg az előző megoldásra utal, de lehet így is:

```
template <class T, class I, class S>
T select(I from, I to) {
    T x = *from++;
    for (; from != to; from++)
        if (S(*from, x)) x = *from;
    return x;
}
```

2. rész:

```
template <class T>
bool Min(T d1, T d2){
  return d1 < d2;
}
```

3. rész:

```
double adat[30];
cout <<select<double>(adat, adat+30, Min<double>);
vagy:
cout <<select<double, double*>(adat, adat+30, Min<double>);
vagy:
cout <<select<double, double*, Min<double> >(adat, adat+30);
```

3. Feladat Σ 5 pont

Egy olyan osztályt (*Vec4)* kell létrehozni, ami dinamikusan változó méretű, valós elemek tárolására alkalmas tömböt valósít meg. A tömb kezdeti mérete, és elemeinek kezdőértéke a konstruktorban adható meg. Meg kell valósítani az alábbi műveleteket:

- *at()* – elem direkt elérése (indexelés). Hibás indexelés esetén `range_error` kivételt dob.
- *insert()* – elemet szúr be a paraméterként megadott hely elé
- *erase()* – kitörli a paraméterként megadott helyen levő elemet
- *size()* – tárolt elemek számát adja
- *count()* – megszámolja, hogy hány olyan elem van a tárolóban, ami azonos a paraméterként kapott értékkel

Az osztály megvalósításán többen dolgoznak egyszerre, akikkel megállapodtak az osztály belső adatszerkezetében, és a tagfüggvények funkcióiban. A megállapodást az alábbi kommentezett deklaráció rögzíti, amin **nem lehet változtatni**. Követelmény, hogy az osztály legyen átadható érték szerint függvényparaméterként, de úgy döntöttek, hogy az **értékadást nem fogják megvalósítani**.

```
class Vec4 {
  Vec4& operator=(const Vec4); // Az értékadó operátor nincs megvalósítva
protected:
  double *v;          // Pointer a dinamikus adatterületre. Ezen a területen tároljuk az adatokat.
  int siz;            // Tárolt elemek száma, azaz ekkora dinamikus területet foglalunk.
public:
  Vec4 (int n = 0, double iv = 0); // Létrehoz egy n elemű vektort, feltölti iv értékkel, inicializál
                      // nulla elemszámhoz is foglal területet, hogy később ne legyen vele baj.
  Vec4 (const Vec4&); // Másoló konstruktor.
  void insert(double d, int i = 0); // A d értéket beszúrja az i. adat elé.
  void erase(int i = 0); // Törli az i. helyen levő adatot.
  int size() const; // Visszaadja a vektorban tárolt elemek számát
  double& at(int);  // Indexelés művelete, ami ellenőrzi, hogy van-e az adott elem, ha nincs hibát dob.
  int count(double d); // Megszámolja, hogy hány d-vel azonos elem van a tárolóban.
  ~Vec4();            // Megszünteti az objektumot
};
```

417. AA Megoldó Alfréd AA 417.

A tagfüggvények elkészítését felosztották egymás között. Önre konstruktorok megírása jutott. **Valósítsa** meg a feladatul kapott tagfüggvényeket! Vegye figyelembe, hogy a mások által írt függvények belső megoldásait nem ismeri, azaz nem használhat ki olyan működést, ami a fenti kódrészletből, vagy annak megjegyzéseiből nem olvasható ki (3 pont).

A Vec4 osztály **módosítása nélkül** készítsen egy *read* függvényt, ami a paraméterként a kapott *std::isteram* típusú adatfolyamról **fájl végéig** érkező valós számokat beolvassa egy szintén **paraméterként kapott** *Vec4* típusú objektumba! (2 pont) Ügyeljen a helyes paraméterezésre! Feltételezheti, hogy az inputon a valós számok whitespace karakterekkel elválasztva követik egymást.

1. részfeladat:

„A" csoport megoldása (kontruktorok):

```
Vec4::Vec4(int n, double iv) :siz(n){
  v = new double[siz];
  for (int i = 0; i < siz; i++)
    v[i] = iv;
}
Vec4::Vec4(const Vec4& f) {
  v = new double[siz = f.siz];
  for (int i = 0; i < siz; i++)
    v[i] = f.v[i];
}
```

„B" csoport megoldása (insert+size):

```
void Vec4::insert(double d, int i) {
  double *tmp = new double[siz + 1];
  int j;
  for (j = 0; j < i; j++)
    tmp[j] = v[j];
  while (j < siz)
    tmp[j+1] = v[j];
  tmp[i] = d;
  delete[] v;
    v = tmp;
    siz++;
}
int Vec4::size() const {
  return siz;
}
```

„C" csoport megoldása (insert+size):

```
void Vec4::erase(int i) {
  double *tmp = new double[siz - 1];
  int j;
  for (j = 0; j < i; j++)
    tmp[j] = v[j];
  for (;j < siz; j++)
    tmp[j] = v[j+1];
  delete[] v;
  v = tmp;
  siz--;
}
int Vec4::size() const {
  return siz;
}
```

„D" csoport megoldása (at+count):

```
double& Vec4::at(int ix) {
  if (ix >= siz)
    throw range_error("indexeles");
  return v[ix];
}
int Vec4::count(double d) {
  int n = 0;
  for (int i = 0; i < siz; i++)
     if (v[i] == d) n++;
  return n;
}
```

2. részfeladat megoldása minden csoportnak:

```
void read(std::istream& is, Vec4& vec) {
  int i;
  while (is >> i)
     vec.insert(i, vec.size());
  vagy:
     vec.insert(i);
}
```

4. Feladat Σ 3 pont

Tételezze fel, hogy **a 3. feladat** *Vec4* osztálya elkészült és hibátlanul működik! Ezen osztály **felhasználásával készítsen** egy olyan *MyVec4* osztályt, ami a *Vec4* osztállyal azonos funkcionalitású, de helyesen kezeli a többszörös értékadást (*s1=s2=s3*) is. (1.5 pont). A megoldást nem értékeljük, amennyiben nem használja a *Vec4* osztályt!
Legyen a *MyVec4* osztálynak egy *instcnt()* tagfüggvénye is, ami megmondja, hogy hányszor példányosították az osztályt. (1.5 pont)

Egy lehetséges megoldás:

```
class MyVec4 :public Vec4 {
  static int cnt;
public:
  MyVec4(int n = 0, double iv = 0) :Vec4(n, iv) {cnt++;}
  MyVec4& operator=(const MyVec4& f);
  static int instcnt() { return cnt; }
};
MyVec4& MyVec4::operator=(const MyVec4& f) {
  if (this != &f) {
    delete[] v;
    v = new double[siz = f.siz];
    for (int i = 0; i < siz; i++)
      v[i] = f.v[i];
  }
  return *this;
}
int MyVec4::cnt = 0;
```

Akik felüldefiniálták a másoló konstruktort és/vagy az értékadást, azok feleslegesen dolgoztak, hiszen a MyVec4 osztály nem kezel dinamikus adattagot, csak a tartalmazott objektum (adat), ami viszont a feltételezésünk szerint jól működik.
Csak akkor kell a másoló konstruktort és/vagy az értékadást felüldefiniálni, ha az alapértelmezés szerinti működés nem jó valamiért!

111. AA Megoldó Alfréd AA 111.

5. Feladat Σ 5 pont

Egy elektronikus irattár tervezésében vesz részt! Az irattárat egy *Mappa* típusú objektum valósítja meg, melyben különböző tartalmú állományok (Allomany) lehetnek. A *Mappa* is egy *Allomany*, így egy tetszőlegesen mély hierarchikus rendszer alakul ki. Az állományok közös attribútuma a név és a hossz bájtban mérve. Jelenleg a következő állományaink vannak, de később ez a felsorolás bővülni fog:

Osztály neve	**egyedi attribútumok**
Mappa	tartalmazott állományok
Szoveg	verzió (egész szám)
Kep	felbontás (2db egész szám)

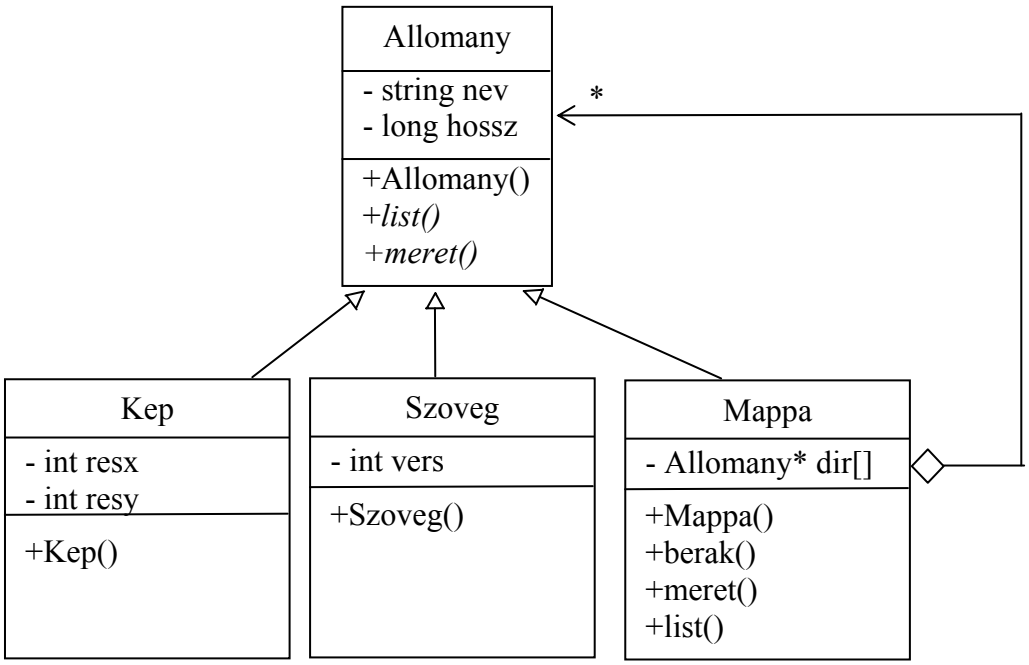
A rendszerben a következő funkciókat kell megvalósítani:

- Állomány katalógusba illesztése. (*berak*)
- Katalógus tartalom (*nevek*) listázása (*list*) (csak az adott szinten levő állományok neveit listázza)
- Mappa és a benne levő minden állomány összesített méretének lekérdezése (*meret*) (rekurzív)

A rendszer tervezéséhez és megvalósításához használhat STL tárolókat is!

Feladatok:

- **Tervezzen** meg egy olyan OO modellt, mely a fenti követelményeket kielégíti! Rajzolja fel a modell osztálydiagramját!
- **Definiálja** az *Allomany*, *Mappa*, *Szoveg*, és *Kep* osztályokat és az elvárt a funkciók ellátásához szükséges tagfüggvényeket! Használja a dőlt betűs neveket!
- **Implementálja** a fenti osztályokat és azok tagfüggvényeit! Az osztályokat olyan módon készítse el, hogy újabb állománytípus felvételekor a már meglevő kódot ne kelljen módosítani! (2 pont)
- **Írjon** egy egyszerű programrészletet, ami készít egy *Mappa* objektumpéldányt, és abba berak egy *Kep*, valamint egy *Szoveg* példányt! (0.5 pont)



A gyakorlatokon ettől bonyolultabb osztályhierarchiát is terveztünk. A második kisZH-ban is bonyolultabb volt!

Azok tanulnak jól, akik megértik, és nem bemagolják ezeket a feladatokat, hiszen nincs két teljesen egyforma feladat.

Rajz: Számonkérések során nem ragaszkodunk a pontos UML szintaxishoz, de elvárjuk, hogy megkülönböztethető módon jelöljék a tartalmazást, az asszociációt, és az öröklést.

```
class Allomany {
    string nev;               // Használjuk az STL string sablonját
    long hossz;
public:
    Allomany(const char *n = "", long l = 0) :nev(n), hossz(l) {}
    virtual void list() { cout << nev; }
    virtual long meret() { return hossz; }
};

class Mappa :public Allomany {
    vector<Allomany*> dir;    // Használjuk az STL vector sablonját
public:
    Mappa(const char *n = "", long l = 0) :Allomany(n, l) {}
    void berak(Allomany *f) {
      dir.push_back(f);
    }
    void list();
    long meret();
};

void Mappa::list() {
    cout << "A katalogus tartalma: " << endl;
    for (vector<Allomany*>::size_type i = 0; i < dir.size(); i++) {
       dir[i]->Allomany::list();
       cout << endl;
    }
}
long Mappa::meret() {
    long sum = Allomany::meret();
    for (vector<Allomany*>::size_type i = 0; i < dir.size(); i++)
       sum += dir[i]->meret();
    return sum;
}

class Szoveg :public Allomany {
    int vers;
public:
    Szoveg(const char *n = "", long l = 0, int v = 1) :Allomany(n, l), vers(v) {}
};

class Kep :public Allomany {
    int resx;
    int resy;
public:
    Kep(const char *n = "", long l = 0, int x = 800, int y = 600) :Allomany(n, l),
resx(x), resy(y) {}
};

...
    Mappa d("/");
    Kep k1("kep1", 100);
    Szoveg t1("text1", 300);
    d.berak(&k1);
    d.berak(&t1);
```