1. BB Megoldó Béla BB 1.

Programozás alapjai 2. (inf.) zárthelyi	2007.05.17. gyakrolat: **G2/V2.706**	Érdemjegy:
	FQ2J73Q **Ch.Max./1.**	Hftest: **4**

Minden beadandó lap tetejére írja fel balra a feladat számát, jobbra a nevét és tankörét! A megoldások során feltételezheti, hogy minden szükséges input adat az előírt formátumban rendelkezésre áll.
A feladatok megoldásához csak a letölthető C és C++ összefoglaló, valamint egy A4-es lapra saját kézzel írt „puska” használható. Számítógép, notebook, menedzser kalkulátor, organiser, rádiótelefon nem használható.

*A feladatokat **figyelmesen olvassa** el! **Ne írjon felesleges** függvényeket ill. kódot, a feladatra koncentráljon! Jó munkát!*

F.	Max.	Elért
1	5	
2	2	
3	3	
4	3	
5	7	
Σ	**20**	

1. Feladat 5 pont

Írjon generikus 2 dimenziós mátrixot! A konstruktor *paraméterként* kapja mátrix méretét (sor, oszlop). Valósítsa meg következő műveleteket:

- *put* – elemet tesz a mátrix megadott indexű (i,j) helyére. Nem létező index esetén dobjon kivételt!
- *get* – elemet vesz ki a mátrix megadott indexű helyéről. Nem létező index esetén dobjon kivételt!
- *size* – visszaadja a mátrixban tárolt elemek számát.

Az osztály:

- legyen átadható érték szerint függvényparaméterként!
- kezelje helyesen a többszörös értékadást (*m1=m2=m3*)!

Írjon egy egyszerű, maximum 10 utasításból álló programrészletet, amely bemutatja az osztály szolgáltatásainak használatát *int* típusra. Mutassa be a kivételkezelést is!

2. Feladat 2 pont

Döntse el, hogy hibásak-e az alábbi programrészletek! Ha igen, adja meg a hiba **helyét** és magyarázatát!
A programrészletekben elírások nincsenek, az összes meghívott függvény ill. hivatkozott osztály létezik, ha nincs megadva a kódrészletben, akkor feltételezze a helyes működést. A memóriaszivárgás is hibának minősül! Válaszait a kódrészletek mellett ill. alatt adja meg!

a)
```
class B { double re2; ...
public:
     friend void operator<<(ostream& os, B& e) { os << e.re2; }
};
...
B b12;
cout << b12 << endl;
```

Hiba oka, ha van:
<< így nem fűzhető, mert nem istream a visszatérési értéke.

b)
```
class B { char *p; ...
     B(const B& b) { f(b); }
     void f(const B b) { ... }
};
```

Hiba oka, ha van:
A másoló konstruktor önmagát hívja, mivel f() hívásakor is meghívódik a másoló -> végtelen ciklus

3. Feladat 3 pont

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

```
#include <iostream>
using namespace std;
class A {
public:
    A(const int i = -1)       { cout << 'k' << i; }
    A(const A& a)             { cout << 'c'; }
    virtual void operator++(int)   { cout << 'p'; }
    ~A()                      { cout << 'd'; }
};
class B :public A {
public:
    B(const char *n = "2j72") { cout << 'K' << n; }
    B(int i) :A(i)            { cout << 'K'; }
    B(const B& b) :A(b)       { cout << 'C'; }
    void operator++(int)      { cout << 'P'; }
    ~B()                      { cout << 'D'; }
};
int main() {
    B b1("2j7");          cout << '\n';
    B b2 = b1;            cout << '\n';
    A* p1 = new B(2);     cout << '\n';
    (*p1)++;              cout << '\n';
    delete p1;            cout << '\n';
    return(0);
}
```

k	-	1	K	2	j	7								
c	C													
k	2	K												
P														
d														
D	d	D	d											

4. Feladat 3 pont

Tervezzen 3 dimenziós vektort reprezentáló osztályt, amely megvalósítja:

- a kivonást a 3d vektorok között,
- a skalárral történő szorzást (balról és jobbról),
- az << operátort, mellyel képes egy *ostream* típusú objektumra (pl: *cout*) kiírni önmagát!

Csak az osztály és az esetleges globális függvények **deklarációja** szükséges. A függvények **definíciója nem** része a feladatnak

4. feladat egy lehetséges megoldása:

```
class Vec {
        double vec[3];
public:
        Vec operator-(const Vec);
        Vec operator*(double);
        friend Vec operator*(double, const Vec&);
        friend ostream operator<<(ostream&, const Vec&);
};
```

5. Feladat 7 pont

Egy meteorológiai állomáson különféle adatokat mérnek (szél, hőmérséklet, csapadék, stb.). A mért adatokat egy OO programmal szeretnénk nyilvántartani, mégpedig úgy, hogy azokat az észlelés sorrendjében egy tárolóba tesszük (*add*). Az adatok eltérő formátumúak. A szélnek pl. iránya és nagysága is van, míg a hőmérséklet csak egy mennyiség, de a sorrend megtartása miatt közös tárolóban kell azokat tárolnunk. A tároló maximum 1000 adatot tud tárolni, és képes kilistázni (*list*) a benne levő adatokat a hozzájuk tartozó értékekkel együtt. (pl. szél: É-Ny, 10km/h; hőmérséklet: 12 C).

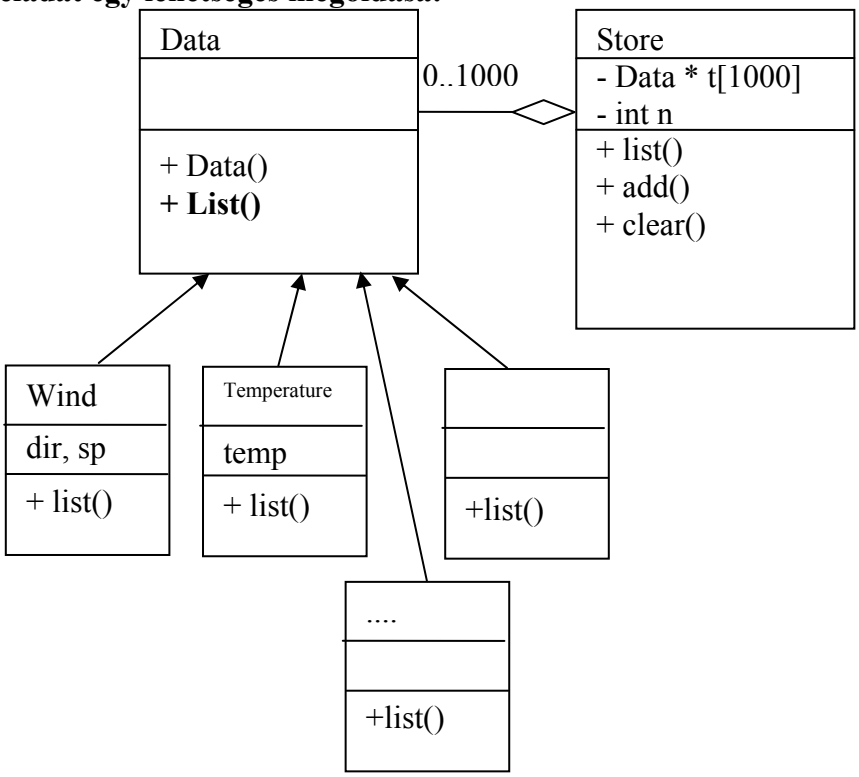
- **Tervezzen** OO modellt, mely tartalmaz *Wind*, *Temperature*, valamint *Store* objektumot, és könnyen bővíthető esetleges újabb adatokkal (pl. csapadék). Rajzolja fel a modell osztálydiagramját! Használja a megadott dőltbetűs neveket!
- **Implementálja** az osztályokat! Ne legyen egy függvénytörzsben sem feleslegest kód! Olyan módon készítse el az osztályokat, hogy bővítéskor ne kelljen a kódot módosítani! A tárolónak legyen olyan művelete is, mellyel az összes adat kitörölhető (*clear*)!
- **Írjon** egy egyszerű programrészletet, ami megmutatja 2 különböző típusú meteorológiai adat bevitelét egy tárolóba, majd kiírja a tárolóban tárolt adatokat és a hozzájuk tartozó értékeket!

1. feladat egy lehetséges megoldása:

```
#ifndef FELADAT1_B
#define FELADAT1_B

// feladat1_b.hpp - generikus 2d matrix sablonja

template <class T>
class Mat {
    T *tp;                          // Elemek tömbjére mutató pointer
    int row;                        // sorok száma
    int col;                        // elemek száma
public:
    Mat(int row, int col) :row(row), col(col) { tp = new T[row*col]; }
    Mat(const Mat&);                // copy konstruktor
    void put(int i, int j, const T&);// elem be
    T get(int i, int j);            // elem ki
    int size() { return row * col; }// elemek száma
    Mat& operator=(const Mat&);     // egyenlőség op.
    ~Mat() { delete[] tp; }         // din. terület felszabadítása
};

template<class T>
Mat<T>::Mat(const Mat& e) {         // copy konstruktor
    tp = 0;                         // operator= híváshoz előkészítjük
    *this = e;                      // most már mehet
}

template<class T>
void Mat<T>::put(int i, int j, const T& e) {
    if (i >= row || j >= col || i < 0 || j < 0)
        throw range_error("Hibas index");
    tp[i*row+j] = e;
}

template<class T>
T Mat<T>::get(int i, int j) {
    if (i >= row || j >= col || i < 0 || j < 0)
        throw range_error("Hibas index");
    return tp[i*row+j];
}
```

```
template<class T>
Mat<T>& Mat<T>::operator=(const Mat& e) { // értékadás operátor
    if(this != &e) {                  // nem önmagának ad értéket, akkor átmásol
        delete[] tp;
        row = e.row;
        col = e.col;
        tp = new T [row*col];
        for (int i = 0; i < row*col; i++)
            tp[i] = e.tp[i];
    }
    return *this;
}
#endif
// programrészlet:
...
#include "feladat1_b.hpp"
...
void main()
{
    try {
        Mat<int>m1(5, 8);
        m1.put(0, 1, 10);
        m1.put(4, 1, 20);
        m1.put(3, 0, 30);
        Mat<int>m2 = m1;
        Mat<int>m3(1, 2);
        m3 = m2;
        cout << m3.get(4,1);
        cout << m2.size();
        cout << m3.get(5,1);
    } catch (exception& e) {
        cerr << e.what();
    }
}
```

5. feladat egy lehetséges megoldása:



```
#include <iostream>
using namespace std;
class Data {
public:
    virtual void list() = 0;
};

class Wind :public Data {
    char    dir[20];        // nem lesz 20-nál hosszabb irány megadva !
    double  sp;
public:
    Wind(char *d, double sp) :sp(sp) {
      strcpy(dir, d);       // ha hosszabb lenne mint húsz, akkor baj lenne
    }
    void list();
};

void Wind::list() {
    cout << "szél: " << dir << sp << " km/h" << endl;
}

class Temperature :public Data {
    double  temp;
public:
    Temperature(double temp) :temp(temp) { }
    void list();
};

void Temperature::list() {
    cout << "hőmérséklet: " << temp << " C" << endl;
}

class Store {
      Data *t[1000];
      int  n;
public:
    Store() :n(0) {};
    void clear();           // adatok törlése
    void add(Data*);        // hozzáadás
    void list();            // adatok listázása
    ~Store() { clear(); }
};

void Store::clear() {
    for (int i = 0; i < n; i++)
        delete t[i];        // töröljük az obj.-t is, mert dinamikusan hoztuk létre
    n = 0;
}
void Store::add(Data *d) {
    t[n++] = d;             // letároljuk a pointert
}
void Store::list() {
    for (int i = 0; i < n; i++)
        t[i]->list();       // meghívjuk az objektumok listázó függvényét
}

main() {
      Store met;
      met.add(new Wind("E-NY", 12));
      met.add(new Temperature(5.3));
      met.list();
      return 0;
}
```