2. labor kis ZH, 2018. március 22., csütörtök 10-12

NÉV (olvasható): NEPTUN kód:

Figyelem, a feladat fizikai szempontból nem feltétlenül helyes, csak a programozási kérdésekre koncentrálj, ne gondolkozz a fizikai realitásán!

Definiálj osztályt anyag tárolására! Az anyagnak van tömege (*m*) és hőmérséklete (*T*), és két további tagváltozója (*Ce*, *U0*), amelyek a belső energia kiszámításában segítenek. Minden tagváltozó legyen elérhetetlen az osztályon kívülről! (1p)

Készíts alapértelmezett konstruktort, amely az m és T tagváltozókat 0-ra állítja, *Ce*-t 4.2-re, *U0*-t pedig 1250-re. Készíts olyan kétparaméteres konstruktort is, amely *Ce* és *U0* értékét a felhasználó által megadott kezdőértékre inicializálja, a másik két tagváltozót 0-ra. (A két konstruktort egyben is megvalósíthatod.) (2p)

Készíts setter függvényeket a *m* és *T* beállításához (2 db függvény), valamint getter függvényekert mind a négy tagváltozó lekérdezéséhez (4 db függvény)! (3p)

Készíts + operátort, amely két anyag összegét számolja ki, új anyagot hoz létre a következő összefüggésekkel: $Ce_{összeg}=(Ce_1*m_1+Ce_2*m_2)/(m_1+m_2)$, $U0_{összeg}=(U0_1*m_1+U0_2*m_2)/(m_1+m_2)$, $m_{összeg}=m_1+m_2$, $T_{összeg}=(T_1*m_1+T_2*m_2)/(m_1+m_2)$. (2p)

Készíts globális függvényt, amely egy anyagot kap paraméterként, és kiszámítja a belső energiáját a következő összefüggéssel: *U0**m+T*Ce*m*. (1p)

Készíts main függvényt, amelyben bemutatod a megírt függvények működését! (1p, ha minden hiányzik, -2p)

max. 10p, megoldási idő: 15 perc

```
#include <cstdio>

class anyag {
    double m, T; // privát: 1p
    double Ce, U0;
public:
    anyag(double ce = 4.2, double u0 = 1250) :m(0), T(0), Ce(ce), U0(u0) {}
    // def ctor: 1 p
    // 2 par ctor: 1p
    double get_m()const { return m; } // getterrek: 2 p
    double get_T()const { return T; }
    double get_Ce()const { return Ce; }
    double get_U0()const { return U0; }
    void set_m(double uj) { m = uj; } // setterek: 1p
    void set_T(double uj) { T = uj; }
    anyag operator+(const anyag & masik)const { // + op: 2p
        double ossz_m = m + masik.m;
        anyag osszeg((Ce*m + masik.Ce*masik.m) / ossz_m, (U0*m + masik.U0*masik.m) / ossz_m);
        osszeg.m = ossz_m;
        osszeg.T = (T*m + masik.T*masik.m) / ossz_m;
        return osszeg;
    }
};

double belso_energia(const anyag & a) { // 1p
    return (a.get_U0() + a.get_T()*a.get_Ce())*a.get_m();
}

int main() {
    anyag viz, cukor(1.2, 360), keverek; // 1 pontról indul, ha valami hiányzik -0,5p/db, elég
egy getter és egy setter hívás
    viz.set_m(1000);
```

```
    viz.set_T(50);
    cukor.set_m(150);
    cukor.set_T(25);
    keverek = viz + cukor;
    printf("Keverek T=%g, m=%g, Ce=%g, U0=%g\n", keverek.get_T(), keverek.get_m(),
keverek.get_Ce(), keverek.get_U0());
    printf("belso energia=%g\n", belso_energia(keverek));
    return 0;
}
```