3. labor kis ZH, 2022. április 27., csütörtök 10-12

NÉV (olvasható): NEPTUN kód:

Készíts osztályt tetszőleges számú valós típusú érték növekvő sorrendben való tárolására (*RendVektor*)! (Dinamikus adatszerkezet használata szükséges.) Az osztály tagváltozóihoz kívülről ne lehessen hozzáférni. Az osztály STL tárolót nem használhat. Készíts

- alapértelmezett konstruktort,
- és másoló konstruktort,
- a tároló tartalmát törlő *clear* tagfüggvényt (eredménye egy üres, de használható tároló),
- destruktort,
- láncolható = operátort,
- elemszámot lekérdező *size* tagfüggvényt,
- [] operátort,
- << operátort, amely szabványos ostream-be írja a tároló tartalmát: először a méretet, utána a tárolt értékeket, szóközzel elválasztva,
- *main* függvényt, amelyben (közvetlenül vagy közvetve) kipróbálod a *RendVektor* összes funkcióját!

max. 10p, megoldási idő: 20 perc

```
#include <iostream>

class RendVektor {
    double* t; // adatszerkezet: 1p
    int n;
public:
    RendVektor() :t{ nullptr }, n{ 0 } {} // 0.5p
    RendVektor(const RendVektor& src) : t{ nullptr }, n{ 0 } { // 1p
        *this = src;
    }
    void clear() { // 1p
        delete[]t;
        t = nullptr;
        n = 0;
    }
    ~RendVektor() { clear(); } // 0.5p
    const RendVektor& operator=(const RendVektor& src) { // 2p
        if (this != &src) {
            if (src.n != n) {
                clear();
                n = src.n;
                t = new double[n];
            }
            for (int i = 0; i < n; i++)
                t[i] = src.t[i];
        }
        return *this;
    }
    int size()const { return n; } // 0.5p
    double& operator[](int i) { return t[i]; } // 0.5p (elég az egyik)
    const double& operator[](int i)const { return t[i]; }
    void add(double uj) { // nem volt feladat
        double* temp = new double[n + 1];
        int i = 0;
        for (; i < n && t[i] < uj; i++)
            temp[i] = t[i];
        temp[i] = uj;
        for (; i < n; i++)
            temp[i + 1] = t[i];
        delete[] t;
```

```
        t = temp;
        n++;
    }
};


std::ostream& operator<<(std::ostream& os, const RendVektor& v) { // fejléc: 1p, törzs: 1p
    os << v.size();
    for (int i = 0; i < v.size(); i++)
        os << ' ' << v[i];
    return os;
}

int main() { // 2p: egyvalami hiányozhat, utána -0.5p/hiányzó
    RendVektor a;
    a.add(3.14);
    a.add(2.71);
    a.add(10.0);
    a.add(-1.0);
    RendVektor b(a); // =-t is hívja, ami a clear-t
    std::cout << b << std::endl; // size-ot és []-t is hívja
    return 0;
}
```