2. labor kis ZH, 2025. március 12., csütörtök 10-12

NÉV (olvasható): NEPTUN kód:

Készíts Intervallum osztályt! Az osztálynak két adattagja van: az intervallum alsó és felső határa, mindkettő valós szám, ezeket az osztályon kívülről elérhetetlenül tárolja. Amelyik tagfüggvény lehet, legyen konstans! A dinamikus memóriakezelés a C++ operátoraival történjen!

- Legyen egy beállító függény, amely két valós számot kap paraméterként. A két szám közül a kisebb legyen az intervallum alsó határa, a nagyobb pedig a felső határa!
- Legyen két lekérdező függvény, amelyek visszaadják az intervallum alsó ill. felső határát.
- Készíts felosztás függvényt, amely paraméterként kap egy pozitív valós értéket (lépésköz), és visszaad egy valós értékeket tároló dinamikus tömböt és a tömb méretét. A két visszaadott érték közül legalább az egyiket referencia paraméterként adja vissza! A függvény feladata, hogy egy új dinamikus tömböt az intervallum alsó határától a felső határáig a megadott lépésközzel haladva töltse fel értékekkel, és ezt adja vissza. (A tömb első és utolsó eleme az alsó ill. a felső határ kell legyen.) Pl. ha a tartomány 1 és 1.5, és a lépésköz 0.2, akkor a kimenő tömb elemei: 1, 1.2, 1.4, 1.5. Ha a lépésköz érvénytelen (nem pozitív), akkor 0 méretet és nullpointert adjon vissza!

Írj main függvényt, ahol kipróbálod a megírt osztály minden függvényét! A lefoglalt dinamikus memóriát ne feledd felszabadítani!

max. 10p, megoldási idő: 18 perc

```cpp
#include <iostream>

class Intervallum { // class szintaxis 1p
    double also, felso; // privát adat 1p
public: // publikus függvények 1p
    void set(double a, double f) { // setter: 1p
        if (a < f) {
            also = a;
            felso = f;
        }
        else {
            also = f;
            felso = a;
        }
    }
    double getA()const { return also; } // getterek, const legyen: 1p
    double getF()const { return felso; }
    double* felosztas(double lepeskoz, int& ujMeret)const { // fejléc: 1p
        if (lepeskoz <= 0) { // hibakezelés: 1p
            ujMeret = 0;
            return nullptr;
        }
        ujMeret = (int)((felso - also) / lepeskoz) + 2; // függvény algoritmusa: 2p
        double* t = new double[ujMeret]; // new[]: 1p
        int i = 0;
        for (double x = also; x < felso; i++, x += lepeskoz)
            t[i] = x;
        t[ujMeret - 1] = felso;
        return t;
    }
};

int main() {
    Intervallum a; // változódef + az összes fvhívás: 2p
    a.set(1, 1.5);
    int n;
    double* t = a.felosztas(0.2, n);
    std::cout << "[" << a.getA() << ", " << a.getF() << "] = ";
    for (int i = 0; i < n; i++)
        std::cout << t[i] << ", ";
    delete[] t; // delete[]: 1p
    return 0;
}
```