4. labor kis ZH, 2025. április 10., csütörtök 10-12

NÉV (olvasható): NEPTUN kód:

Adott az alábbi Complex osztály.

- Egészítsd ki úgy, hogy az osztály tartsa nyilván, hogy a program futása során hány példány jött létre belőle! (Globális vagy publikus változót nem használhatsz.) Az ehhez szükséges változót és a szükséges programkódot is add meg! (Más konstruktort nem kell írnod.)
- Írj megfelelő függvényt, amellyel akkor is lekérdezheted, hogy hány Complex példány jött létre a program futása során, ha nincs Complex típusú objektum.
- Írj láncolható beolvasó operátort, amely segítségével egy Complex értéket be tudsz olvasni a szabványos bemenetről, vagy azzal kompatibilis más bemenetről! A valós és a képzetes rész egy szóközzel elválasztott valós értékpár formájában érkezik.
- Írj main függvényt, ahol létrehozol egy Complex értékek tárolására alkalmas vector-t, és a szabványos bemenetről beolvasott Complex értékekkel feltöltöd. A beolvasáshoz használd az előző pontban létrehozott beolvasó operátort! A beolvasás végét olyan Complex érték jelzi, amelynek valós és képzetes része is 0. Ezt az értéket már ne tárold a vectorban!
- Másold át egy másik vector-ba azokat az értékeket, amelyeknek a valós része 0.0!
- Írd ki, hogy a program futása során eddig hány Complex példány jött létre!

```
class Complex {
	double re, im;

public:
	Complex(double Re = 0.0, double Im = 0.0) : re(Re), im(Im) {

	}
	double getRe()const { return re; }
	double getIm()const { return im; }
	void setRe(double Re) { re = Re; }
	void setIm(double Im) { im = Im; }
	bool is0()const { return re == 0.0 && im == 0.0; }

};
```

max. 10p, megoldási idő: 20 perc

```cpp
#include <iostream>
#include <vector>

class Complex {
	double re, im;
	static int db; // 1p
public:
	Complex(double Re = 0.0, double Im = 0.0) : re(Re), im(Im) {
		db++; // 1p
	}
	double getRe()const { return re; }
	double getIm()const { return im; }
	void setRe(double Re) { re = Re; }
	void setIm(double Im) { im = Im; }
	bool is0()const { return re == 0.0 && im == 0.0; }
	static int getDb() { return db; } // 1p
};

int Complex::db = 0; // 1p

std::istream& operator>>(std::istream& is, Complex& c) { // 1p
	double re, im; // működés: 1p
	is >> re >> im;
	c.setRe(re);
	c.setIm(im);
	return is; // 1p
}

int main() {
	std::vector<Complex> v1; // 1p
	Complex x;
	std::cin >> x; // 1p
	while (!x.is0()) { // 1p
		v1.push_back(x); // 1p
		std::cin >> x;
	}
	std::vector<Complex> v2;
	for (size_t i = 0; i < v1.size(); i++) // az egész másoló rész: 2p
		if (v1[i].getRe() == 0.0)
			v2.push_back(v1[i]);
	std::cout << Complex::getDb(); // 1p
	return 0;
}
```