Języki programowania, czwartek 20.11.2014
Zadanie 6

Należy stworzyć klasę `ExampleClass`, mającą demonstrować różne mechanizmy języka C++.

Klasa powinna zawierać pola składowe:
- `double n;` // przechowującą liczbę rzeczywistą
- `double* pointer;` //przechowującą wskaźnik na liczbę rzeczywistą
- `double* array;` //zawierającą tablicę liczb rzeczywistych alokowaną dynamicznie
- `int fSize;` //ilość elementów tablicy `pointer`
- `int fCount;` //zdefiniowane jako pole statyczne zliczające ilość obiektów typu `ExampleClass`

dwa konstruktory:
- domyślny – ustawia pole `n` oraz wartość wskaźnika `pointer` na 0, rozmiar tablicy `array` na 10, alokuje jej pamięć oraz ustawia wszystkie 10 elementów na wartość 1. Użycie konstruktora zwiększa pole statyczne o 1
- kopiujący – tworzy kopię danego obiektu (ustawia wszystkie pola nowego obiektu na wartości takie same jak obiektu kopiowanego)

destruktor:
- usuwa wskaźnik `pointer`, tablicę `array` oraz zmniejsza wartość pola statycznego o 1 //**1.5 pkt.**

metody:
- `void SetN(double N)` – ustawia pole `n`
- `void SetPointer(double val)` – ustawia wartość wskaźnika `pointer`
- `void Print()` - wypisuje na ekran pole `n`, wskaźnik `pointer` oraz tablicę `array`
- `int GetCount()` - metoda statyczne, wypisuje ilość elementów typu `ExampleClass` //**0.5 pkt.**

w programie należy zaimplementować przeciążanie:
- `operator+` - dodaje wartości `n` i `pointer` do siebie z dodawanych obiektów, tablica `array` wynikowego obiektu powinna mieć 10 elementów ustawionych na wartość 2 //**0.5 pkt**

Jeżeli zdążymy, przeciążamy kolejne operatory (jeśli nie, jest to **obowiazkowe** zadanie domowe):
- `operator=` - przypisuje obiektowi z lewej strony znaku parametry obiektu stojącego z prawej strony znaku
- `operator[]` - zdefiniowany tak, by mógł stać po obu stronach znaku =, użycie z wartością 1 wypisuje pole `n`, z wartością 2 wypisuje wartość wskaźnika `pointer`, z dowolną inną wypisuje ostatni element tablicy `array`
- `operator<<` - działa identycznie jak metoda `Print()` (ale jej nie używa) //**2 pkt.** (ten operator implementujemy zawsze jako funkcję zaprzyjaźnioną z klasą, pozostałe możemy implementować jako metody składowe lub funkcje zaprzyjaźnione)

Zaimplementowanie operatorów (oprócz <<) zarówno jako metod składowych klasy jak i funkcji zaprzyjaźnionych //**0.5 pkt.**

W programie głównym należy przetestować działanie wszystkich elementów klasy `ExampleClass`.

Do kompilacji używamy pliku **Makefile** (będzie wymagany na kolokwium).