Języki programowania+, Zadanie 4

Zadanie ma na celu dalsze przećwiczenie tworzenia konstruktorów, destruktorów, a także metod i pól statycznych w klasach. Program demonstruje też wykorzystanie biblioteki STL języka C++. Do tego stworzymy klasę `Ksiazka`.

Należy stworzyć klasę `Ksiazka`, która będzie zawierać pola:
`char *tytul`,
`std::string autor`,
`enum gatunek (przygodowa=1, lektura=2, fantastyka=3, biografia=4)`,
oraz pole statyczne `int fIlosc`.
Klasa powinna zawierać również dwa konstruktory:
`Ksiazka()` - konstruktor bez parametrów (ustawia pole `tytul` na "nazwa" oraz `autor` na "`autor`"),
`Ksiazka(char* tyt, char* aut)` – konstruktor z parametrami, konstruktor kopiujący
oraz destruktor. Każde użycie dowolnego konstruktora zwiększa ilość książek o 1, każde użycie destruktora zmniejsza ilość książek o 1.
Oprócz tego, należy stworzyć metody "set" i "get" do każdego z pól i statyczną metodę zwracającą pole statyczne `fIlosc`.

Część I
Do wykonania:

1. Stworzenie klasy `Ksiazka` i użycie jej w programie (stworzenie obiektu, użycie wszystkich konstruktorów, wypisanie elementów składowych – przy użyciu metod "get", zmiana parametrów przy użyciu metod "set" i ich ponowne wypisanie). **2 p.**
2. Stworzenie wskaźnika na obiekt `Ksiazka`, użycie konstruktora wraz z operatorem `new` i usunięcie obiektu operatorem `delete`. **1 p.**

W programie należy trzy razy użyć metody statycznej do wypisania ilości książek: na samym początku, po stworzeniu wszystkich obiektów i wskaźników, oraz po usunięciu części obiektów przy użyciu delete.

W drugiej części zadania będziemy chcieli stworzyć listę obiektów typu `Ksiazka`. Do tego celu wykorzystamy listę podwójnie linkowaną (double-linked list) z tzw. Standardowej Biblioteki Szablonów języka C++ (STL). Przykład ten pokazuje możliwości wykorzystania gotowych algorytmów i struktur danych dostępnych w języku C++.
Użycie listy z biblioteki standardowej pokazuje przykład poniżej:

```
#include <string>
#include <list>
#include <iostream>

using namespace std;

class A
{
public:
  string fCiagZnakow;
  int fLiczba;
};

int main()
{
  A a1;
  A a2;

  list<A> lista; //lista obiektow klasy A
```

```
  //wstawianie elemmentow na koniec listy
  lista.push_back(a1);
  lista.push_back(a2);

  //iterowanie po elementach listy
  list<Jablko>::iterator i;
  for(i=lista.begin();i!=lista.end();i++)
    {
      cout<<(*i).fCiagZnakow<<endl;
      cout<<(*i).fLiczba<<endl;
      cout<<endl;
    }

  //usuwanie elementu z konca listy
  lista.pop_back();

  //zwrocenie elementu z konca listy
  A a3 = lista.back();

  return 0;
}
```

Część II

Do wykonania:

1. Zadeklarować listę obiektów typu `Ksiazka`, wstawić minimum 3 elementy (deklarując je wcześniej) i wypisać listę iterując po nich, po czym usunąć ostatni element (użyć metody statycznej do wypisania ilości książek). **1 p.**
2. Tworzymy listę wskaźników na obiekt `Ksiazka`, wstawić do listy trzy wskaźniki: jeden przez referencję do obiektu, który stworzyliśmy na początku zadania, drugi tworząc wskaźnik i alokując mu pamięć operatorem `new` i korzystając z konstruktora, trzeci, wstawiając bezpośrednio w metodzie `push_back` operator `new` oraz konstruktor, np. w podobny sposób:
   `lista.push_back(new A());`
   Następnie usuwamy ostatni element z listy i wypisujemy ilość elementów danego typu oraz ponownie iterujemy po liście. Dlaczego liczba elementów jest inna niż ilość elementów w liście (przedyskutować z prowadzącym)? Tworzymy wskaźnik na obiekt i pobieramy ostatni element z listy metodą `back` (ustawiamy to przed usunięciem ostatniego elementu!). Następnie, po usunięciu ostatniego elementu z listy, używamy na tym wskaźniku operatora `delete` i ponownie sprawdzamy ilość obiektów danego typu. **1 p.**

Uwagi:

1. Do kompilacji używamy kompilatora g++:
   `g++ -Wall main.cpp klasa1.cpp klasa2.cpp -o main`
1. Strony z dokumentacją oraz przykładami:
   http://www.cplusplus.com/reference/stl/list/
   http://www.yolinux.com/TUTORIALS/LinuxTutorialC++STL.html