Języki programowania, Zadanie 4

Zadanie ma na celu przećwiczenie użycia wielu klas, tworzenia konstruktorów, destruktorów, metod i pól statycznych w klasach. W tym celu stworzymy klasę `Book`.

Należy stworzyć klasę `Book`, która będzie zawierać następujące pola:
`char *title`,
`std::string author`,
`int pages`,
oraz pole statyczne `int fCount`.
Klasa powinna zawierać również dwa konstruktory:
`Book()` - konstruktor bez parametrów (ustawia pole `title` na "tytul", author na "autor" oraz `pages` na 0),
`Book(char* Title, string Author, int Pages)` – konstruktor z parametrami, konstruktor kopiujący oraz destruktor. Każde użycie dowolnego konstruktora zwiększa ilość książek o 1, każde użycie destruktora zmniejsza ilość książek o 1. Należy zachować wszystkie dobre praktyki programowania (przypominam o zwalnianiu pamięci).
Oprócz tego, należy stworzyć metody "set" i "get" (jeśli potrzebne) i statyczną metodę zwracającą pole statyczne `fCount`.

Do wykonania:

1. Stworzenie 3 obiektów klasy `Book` i użycie ich w programie (stworzenie 3 obiektów poprzez użycie wszystkich konstruktorów, wypisanie elementów składowych, zmiana tytułu, itp.). **1 p.**
2. Stworzenie 3 wskaźników obiektów `Book` poprzez użycie 3 konstruktorów wraz z operatorem `new` i usunięcie obiektów na koniec programu operatorem `delete`. **1 p.**

Po każdym stworzeniu i usunięciu obiektu używamy metody statycznej do wypisania ilości książek.

W drugiej części zadania będziemy chcieli stworzyć dynamicznie alokowaną tablicę wskaźników obiektów typu `Book`. Do tablicy wstawiamy 4 elementy – 3 wskaźniki utworzone wcześniej oraz wskaźnik poprzez użycie `new` bezpośrednio na elemencie tablicy (i oczywiście wypisujemy na ekran elementy tablicy). **1 p.**
Następnie będziemy chcieli stworzyć listę obiektów typu `Book`. Do tego celu wykorzystamy listę podwójnie linkowaną (double-linked list) z tzw. Standardowej Biblioteki Szablonów języka C++ (STL). Przykład ten pokazuje możliwości wykrorzystania 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<A>::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 `Book`, wstawić minimum 3 elementy (deklarując je wcześniej) i wypisać listę iterując po nich, po czym usunąć ostatni element (użyć metody statycznej do wypisania ilości trójkątów). **1 p.**
2. Tworzymy listę wskaźników na obiekt `Book`, 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. na końcu, po usunięciu ostatniego elementu z listy, używamy na tym wskaźniku operatora `delete` i ponownie sprawdzamy ilość obiektów danego typu. **1 p.**

Do kompilacji używamy `Makefile.`